UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA DE TELECOMUNICACIÓN

PROYECTO FIN DE CARRERA

ANALISIS Y COMPARATIVA DE LOS PROTOCOLOS DE TRANSMISIÓN DE VÍDEO ADAPTATIVO POR INTERNET

AUTOR: VÍCTOR MATA GALIANO

TUTOR: ANDRÉS MARÍN LÓPEZ

Leganés, 2014

Agradecimientos

En primer lugar tengo que agradecer el apoyo de mis padres durante estos años, ya que sin ellos probablemente no habría llegado a acabar esta carrera. Por sus ánimos en los momentos más difíciles, son la parte más importante de mi vida.

Luego agradecer al resto de mi familia, a mi hermano Pablo, a mis abuelos Petra y Pedro por lo que me han dado todos estos años, y acordarme de mi Yayo que ya no está y mi Yaya que aunque no puedan ver que me convierto en ingeniero seguro que estarán orgullosos.

Por supuesto que tengo que agradecer a mi tutor, Andrés Marín, por acoger este proyecto, y a Raul García que ha sido una ayuda importantísima, así como a Jose Manuel, Ana Isabel y Enrique Quilis, todos ellos de la empresa en la que he dado mis primeros pasitos.

Citar también a mis compañeros de Cabeceras y todos los compañeros de comidas, Irene, Jaime, Eve,…

Como no a mis amigos y compañeros de la universidad, que me han hecho la estancia en ella más enriquecedora, sobre todo a David (mi primer compañero de prácticas y un apoyo constante), a Javi (muchas horas compartidas de estudio y trabajos), a Paloma (por ayudarme siempre que lo he necesitado), a Adri (por estar ahí siempre) y a Abel (otro gran amigo y compañero).

Y a amigos de toda la vida, son pocos pero tengo la certeza de que estarán ahí siempre, por eso quiero destacar a mi amigo Jaime y a Héctor.

Además, quiero agradecer a toda mi gente del Real Aranjuez su trato porque el compaginar ese trabajo con mi carrera universitaria ha sido muy importante para mí. Me gustaría citar a mis amigos Víctor Sánchez, Juanlu, mis “hermanos” Melli, al míster Curro, Raúl, etc.

Seguramente me deje a personas, pero éstas han sido de las más importantes que me han dado lo que hace falta para poder superar los retos que se van presentando. Gracias.

Resumen

La TV por Internet es ya una realidad. Cada vez son más los usuarios que utilizan sus dispositivos móviles para ver algún programa de TV y cada vez son más los operadores que ofrecen contenido "*live*" para retransmitir los eventos más importantes de la actualidad. Los móviles, los iPad/tablets o los ordenadores portátiles van ganando terreno a la TV tradicional.

En este Proyecto Fin de Carrera se realiza un estudio de los cuatro protocolos más importantes actualmente para ofrecer vídeo por Internet de forma adaptativa: Smooth Streaming, HLS, HDS y MPEG-DASH (éste último aún en fase de desarrollo). Todos ellos están basados en el protocolo HTTP.

El proyecto abarca una serie de pruebas en un entorno de laboratorio con propiedades "*live*" que nos permitirá realizar una comparativa entre los protocolos y aprender su modo de funcionamiento.

El objetivo es ofrecer al usuario la mejor calidad posible adecuándose al dispositivo que utiliza y el entorno en el que esté. Aún hay mucho margen de mejora por delante, esto es solo el comienzo.

Palabras clave: adaptativo, CDN, HLS, HDS, HTTP, Smooth, segmentos, DVR, Manifest, stream, calidades, propiedad, reproductor.

Abstract

TV via the Internet is already a fact. There are more and more users using their mobile devices to watch TV programmes and also there are progressively more operators that offer “live” content in order to broadcast the most important current events. Mobile phones, tablets or laptops are catching up with traditional TV.

In this Final-Year Proyect we analyse the currently four most important adaptive streaming video protocols: Smooth Streaming, HLS, HDS and MPEG-DASH (the last one still in development phase). All of them are based on the HTTP protocol.

The Proyect contains various tests in a laboratory environment with “live” properties which will allow us to make a comparison between the different protocols and learn their operation modes.

The aim is to offer the user the best possible quality depending on the device and environment in use. There are lots of issues to improve upon, but this is just the beginning.

Keywords: adaptive, CDN, HLS, HDS, HTTP, Smooth, chunks, DVR, Manifest, stream, qualities, property, player.

Índice General

Índice de figuras

Índice de tablas

1. Introducción

1.1. Motivación del proyecto

Hoy en día, Internet ha evolucionado tanto que está presente en la vida diaria de la mayoría de las personas. La combinación entre Internet y la TV cada vez tiene más importancia para los proveedores de televisión. Los usuarios no siempre pueden ver en directo un contenido/programa o no están en casa para poder ver con la televisión tradicional una serie, un programa deportivo o las noticias. Además, cada vez existen más dispositivos móviles (smartphones, tablets, portátiles,…) con conexión a Internet que nos permiten ver contenido de TV. Este avance ha permitido que un usuario pueda ver un programa en cualquier momento y lugar.

Se lleva ya unos años trabajando para poder ofrecer TV por Internet, tratando cada vez de ofrecer la mejor calidad posible al usuario. Hay que tener en cuenta que cada dispositivo tiene diferentes características, tanto físicas, es decir, tamaño o resolución de la pantalla, cómo de servicio, que tipo de conexión a Internet tiene contratada, ancho de banda o cuál es el estado de la red en el momento de la reproducción del vídeo. A todo esto, habría que añadir la tecnología que se utiliza en cada dispositivo (Apple, Android, Microsoft). Por tanto, debido a esta variedad de dispositivos, existen diferentes protocolos para la transmisión de vídeo por Internet.

Pero se ha querido dar un paso más allá, enfocando el esfuerzo en ofrecer al usuario un contenido con la máxima calidad posible y sin cortes. Y con ese objetivo, han surgido los protocolos de transmisión de vídeo por Internet adaptativos. No todos los usuarios pueden reproducir un vídeo con la misma calidad, sin embargo, tenemos que ser capaces de codificar un contenido a diferentes calidades para que un usuario lo pueda reproducir sin ningún tipo de problema.

Prácticamente todos los proveedores de televisión ofrecen o están comenzando a ofrecer el contenido de sus canales a través de Internet. Para ello hace falta una gran infraestructura que te proporcione soporte y mantenimiento. La idea es ofrecer al usuario tanto contenido en directo ("*live*") como contenido *on demand* (para poderlo ver en cualquier instante).

Por tanto, necesitamos disponer de una amplia red de servidores que nos permitan cachear el contenido para en el momento en que el usuario decida acceder a él, se le pueda proporcionar de manera rápida y con la mayor de las calidades posibles.

A día de hoy, hay tres protocolos de transmisión de vídeo adaptativo dominantes, cada uno de ellos asociado a una tecnología y apropiado para dispositivos diferentes.

Uno de los pioneros fue el protocolo de Smooth Streaming, también denominado HSS (HTTP Smooth Streaming), la solución de Microsoft; además tenemos HLS (HTTP

Live Streaming) la solución de Apple y HDS (HTTP Dynamic Streaming) la solución de Adobe.

Por otro lado, existe un protocolo más reciente denominado MPEG-DASH (Moving Picture Experts Group-Dynamic Adaptive Streaming over HTTP) del que se espera sea la solución del futuro cercano (en estos próximos años), logrando aunar las diferentes tecnologías y pudiendo ser útil para todos los dispositivos.

La motivación es poder ofrecer al usuario el contenido que éste quiera ver en el momento y lugar que elija con las mejores prestaciones posibles.

1.2. Objetivos

El objetivo principal de este proyecto es realizar un estudio de los cuatro protocolos más importantes de transmisión de vídeo adaptativo por Internet y realizar un análisis comparativo de los mismos dentro de un marco real, es decir, a través de una red completa que nos permita tanto generar como distribuir el contenido de vídeo hasta el usuario final.

A continuación enumeraremos una serie de objetivos que se persiguen mediante la realización de este proyecto:

- Llevar a cabo un estudio teórico de cada uno de los protocolos de transmisión de vídeo adaptativo analizando su estructura y su modo de funcionamiento.
- Definir los conceptos de CDN y propiedad.
- Utilizar una estructura de red para la transmisión de vídeo adaptativo por Internet.
- Solicitar al proveedor de servicios de CDN distintas propiedades asociadas a cada uno de los protocolos de transmisión de vídeo estudiados.
- Realizar pruebas en un entorno de laboratorio empleando distintos dispositivos según el protocolo a utilizar.
- Simular situaciones reales de cortes en la señal de vídeo con un dispositivo para generar imperfecciones.
- Analizar el comportamiento de distintos reproductores, tanto de proveedores de televisión como players estándar del protocolo.
- Sacar conclusiones acerca de la robustez de cada uno de los protocolos ante pérdidas de la señal de vídeo.

1.3. Contenido de la memoria

En primer lugar, se realiza un estudio del estado del arte de los protocolos de transmisión de vídeo adaptativo por Internet, con una previa introducción a los métodos de streaming tanto tradicionales como los que nos serán objeto de estudio (todo englobado en el capítulo 2). En este mismo apartado, realizaremos una serie de definiciones clave para entender el entorno en el que llevaremos a cabo el proyecto. Este punto estará a su vez separado en cuatro grandes bloques, uno por cada uno de los protocolos a estudiar (Smooth Streaming, HLS, HDS y MPEG-DASH).

A continuación, en el capítulo 3 se definirán los distintos escenarios para la parte experimental o práctica del proyecto y se documentarán las pruebas realizadas dividiéndolas en cuatro grandes bloques, una por cada uno de los protocolos.

En el capítulo 4, realizaremos un análisis comparativo entre los cuatro protocolos estudiados teniendo en cuenta las pruebas realizadas, extraeremos las principales conclusiones del proyecto y propondremos los posibles caminos o líneas futuras del mismo.

En el capítulo 5, se presentará un presupuesto del proyecto teniendo en cuenta los distintos dispositivos utilizados durante el mismo.

2. Estado del arte

2.1. Introducción.

En los últimos años la utilización de Internet en dispositivos móviles se ha expandido de forma notable y, cada vez es más gente la que utiliza los smartphones, las tablets o los portátiles (entre otros dispositivos) para ver la TV vía Internet. Diversos estudios (realizados por Cisco) [1] indican que para el año 2018 el vídeo por Internet alcanzará el 69,1% del tráfico de datos en dispositivos móviles. Del total de datos que se moverán por las red en un mes, hasta 15,9 exabytes (1 exabyte corresponde a cerca de 109 gigabytes), se estima que 11 exabytes correspondan a transmisión de vídeo. Pero ya hoy en día, el tráfico correspondiente al vídeo supera el 50% del intercambio de datos en la red. Las grandes empresas de telecomunicaciones están apostando por la transmisión de vídeo por internet, y la vía que se está comenzando a utilizar es el objeto de estudio de este proyecto, los protocolos de transmisión de vídeo adaptativo por Internet.

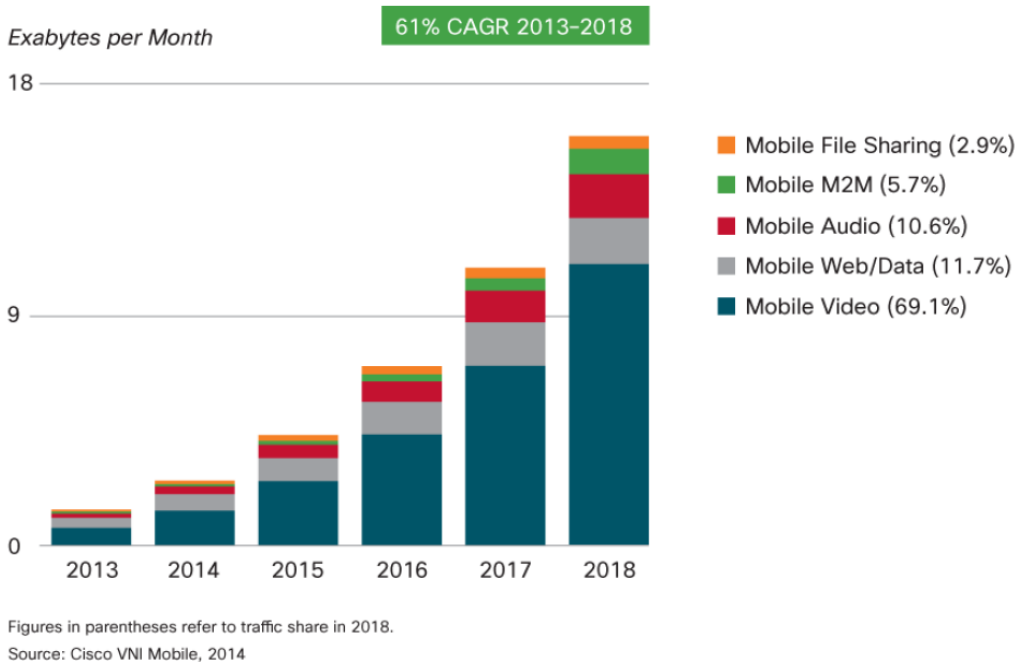


Figura 1. Evolución del tráfico de datos en dispositivos móviles.

El estudio del proyecto se va a centrar en 4 protocolos. La solución de Microsoft "Smooth Streaming", la solución de Apple "HLS" (HTTP Live Streaming), la solución de Adobe "HDS" (HTTP Dynamic Streaming) y MPEG-DASH (Moving Picture Experts Group-Dynamic Adaptive Streaming over HTTP) que trata de aunar todas las tecnologías.

2.1.1. Historia del Streaming

Existen tres formas de transmitir el vídeo por Internet: el Streaming tradicional, la descarga progresiva y el Streaming adaptativo (en el que centraremos nuestro estudio).

2.1.2. Streaming Tradicional

Un buen ejemplo del Streaming tradicional es el protocolo RTSP (Real-Time Streaming Protocol). Es un protocolo con estado, lo que quiere decir que desde que el cliente establece la conexión con el servidor, éste hace un seguimiento de la misma (conoce el estado del cliente) hasta que se cierra dicha conexión. Durante dicha conexión, el servidor envía paquetes RTP de tamaño típico 1452 bytes, que pueden llegar a contener unos 11 milisegundos de vídeo empleando una tasa de codificación de 1 Mbps. Los paquetes se transmiten sobre UDP o TCP, lo que puede provocar problemas con los firewall o proxies en el caso de paquetes UDP o retardos en el caso de TCP.

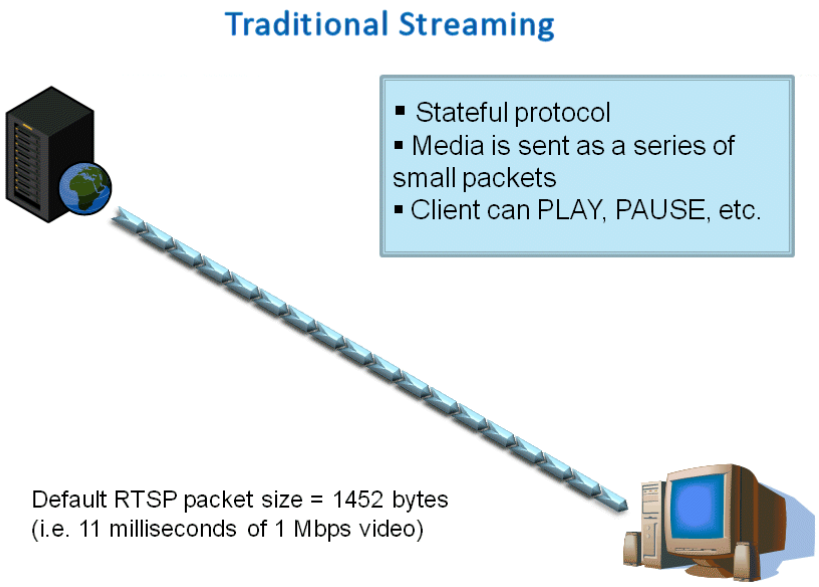


Figura 2. Protocolo RTSP de Streaming Tradicional

Este tipo de protocolos de Streaming tradicional se caracterizan y diferencian del resto de técnicas por:

- El servidor envía paquetes al cliente a una tasa fija, es decir, el bit-rate es aquel con el que se ha codificado el vídeo y no cambia.
- Además, el servidor sólo envía paquetes hasta que se llena el buffer del cliente. El buffer suele comprender un espacio entre 1 y 10 segundos de vídeo. El cliente que utiliza Microsoft, el reproductor de Silverlight [2], tiene un buffer de 5 segundos (por defecto).

Por tanto, en el caso del streaming tradicional, la tasa o bit-rate de codificación debe ser menor que el ancho de banda disponible del usuario si no queremos que la reproducción se corte de forma frecuente.

2.1.3. Descarga progresiva

Esta técnica se basa en la descarga de ficheros de datos de un servidor HTTP. La descarga es progresiva ya que el vídeo se puede reproducir mientras la descarga está activa. Esta técnica se utiliza en muchos sitios web para la descarga de vídeo como YouTube, Vimeo o MySpace.

Si paramos la reproducción de una descarga progresiva y esperamos, la descarga continúa y el contenido se almacena en caché. Posteriormente podremos ver el vídeo completo sin cortes.

El inconveniente que surge es que si finalmente no queremos ver el contenido se habrá gastado el ancho de banda de forma innecesaria al haber descargado todo el contenido.

2.1.4. Streaming Adaptativo basado en HTTP

La principal diferencia del Streaming adaptativo respecto a la descarga progresiva es, que en este caso, se realizan pequeñas descargas progresivas, en lugar de realizar una única descarga de todo el fichero o contenido en su conjunto.

El contenido de vídeo y audio se trocea en pequeños segmentos (chunks) y se codifica según el formato en el que se quiera transmitir. Cada segmento es independiente del anterior y del posterior, por lo que pueden ser decodificados de forma independiente. Los segmentos se guardan en un servidor web HTTP. Cuando un cliente solicita ese contenido, se realiza una descarga progresiva de cada segmento. Según el cliente va descargando los chunks, se reproduce la secuencia de segmentos en orden.

Lo bueno de este método es que es adaptativo, el vídeo/audio se codifica a diferentes calidades (bit-rates), generándose múltiples segmentos para las diferentes calidades. El cliente puede elegir entre las distintas calidades, y en función de los cambios, tanto en la red como en el dispositivo que se utilice, se pueden producir cambios en la calidad del contenido de manera adaptativa.

Nuestro proyecto se basará en la técnica del Streaming adaptativo.

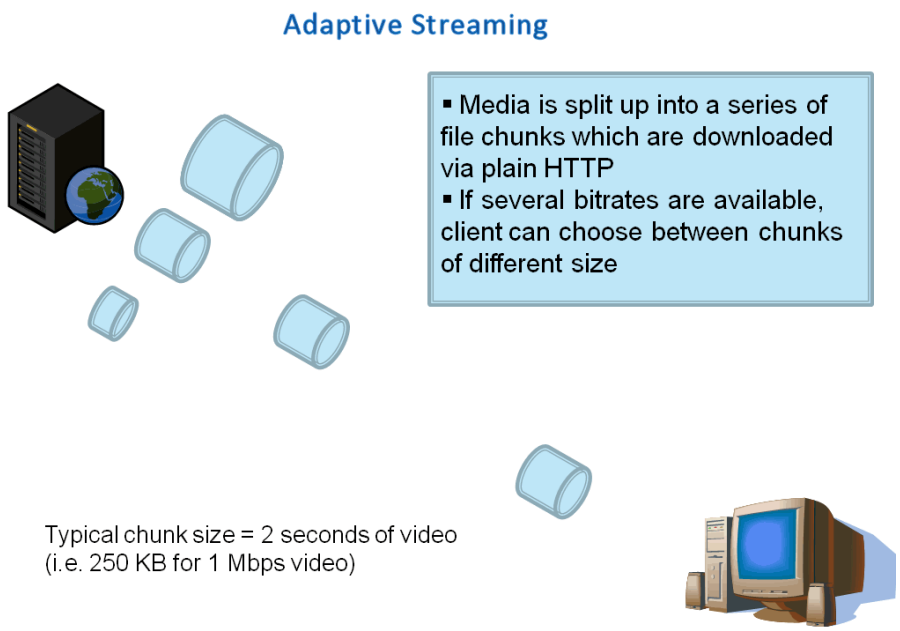


Figura 3. Método de Streaming Adaptativo

2.2. *Utilización HTTP*

De los protocolos de streaming clásicos (RTSP, RTMP) se ha pasado a utilizar HTTP (protocolo sin estado).

Cuando un cliente solicita un contenido, el servidor responde enviando el contenido solicitado pero no recuerda el estado del cliente, es decir, termina la sesión hasta que un cliente vuelva a solicitar algún dato.

El Streaming adaptativo está basado en el protocolo HTTP y presenta las siguientes ventajas:

- Provee múltiples versiones del mismo contenido codificado a diferentes tasas (bit rates) para eliminar problemas de buffering.
- Desde el lado del cliente (y no desde el servidor como se hacía en el Streaming clásico) se determina el bit-rate acorde al ancho de banda disponible, es decir, es el cliente el que selecciona el "*profile*" (perfil) del contenido.

Como ya hemos comentado, el contenido de vídeo se transcodifica a distintas tasas (bit-rates), lo que se conoce como los distintos perfiles ("profiles") del contenido. Cada servicio puede tener hasta un total de 16 profiles que se diferencian en el bit-rate, la resolución o el tipo de codificación. Posteriormente ese contenido de vídeo se divide en fragmentos, chunks o segmentos.

El proceso es el siguiente:

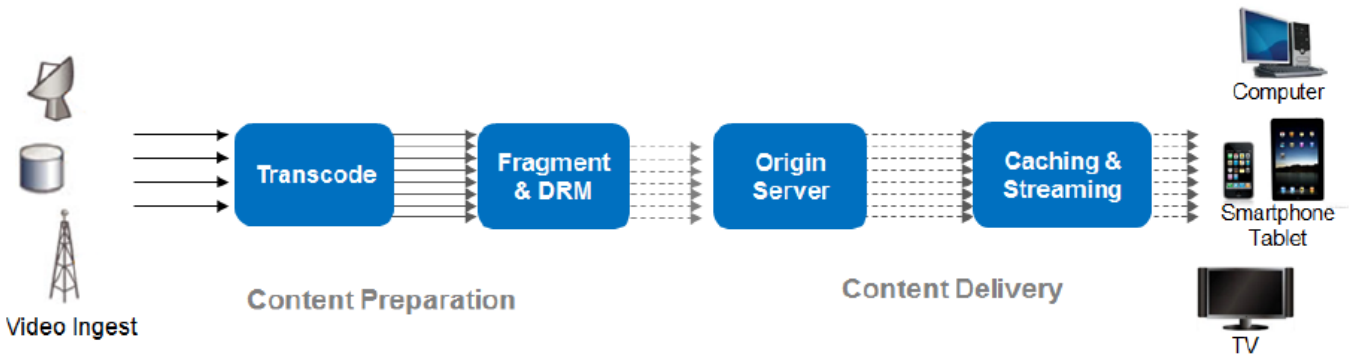


Figura 4. Proceso de transmisión de vídeo

En el *transcoder* se hacen diferentes copias del contenido con diferentes bit-rates, lo que origina los distintos perfiles. Cada copia o perfil se caracteriza por una resolución y un bit-rate determinado. Dependiendo del dispositivo utilizado se tendrán más o menos perfiles.

Network	Device	Profile: Bit rate	Resolution
3G Mobile (3 profiles)	Phone	3G-Low: 100kbps	320x180
		3G-Medium: 250kbps	320x180
		3G-High: 350kbps	416x240
4G (3G profiles + 1)	Phone	4G-High: 650kbps	640x360
WiFi (5 profiles)	Smart Phone / Pad / PC	WiFi-ulow:350kbps	320x180
		WiFi-low:500kbps	416x240
		WiFi-mid: 850kbps	640x360
		WiFi-high: 1.5mbps	640x360
		WiFi-uhigh:2.5mbps	1280x720
Broadband (8 profiles)	Pad / PC / Console / STB	350 Kbps	320x180
		150 Kbps	320x180
		500 Kbps	416x240
		1.0 Mbps	640x360
		750 Kbps	640x360
		1.25 Mbps	864x486
		1.5 Mbps	960x540
		3 Mbps	1280x720

Tabla 1. Perfiles de vídeo de distintos dispositivos

Posteriormente el contenido se fragmenta en pequeños ficheros con una duración entre 2 y 10 segundos dependiendo del protocolo de transmisión de vídeo a utilizar. Además, se puede aplicar un sistema de protección del contenido denominado DRM (Digital Rights Management).

PROTOCOL	TYPICAL SEGMENT DURATION
MPEG-DASH	Flexible
HLS	10 sec
HDS	2 – 4 sec
SMOOTH STREAMING	2 – 4 sec

Tabla 2. Duración recomendable de los segmentos según el protocolo

En los servidores se encuentran los ficheros *Manifest* dónde se indican las diferentes calidades de un contenido.

A continuación podemos analizar el comportamiento del cliente al utilizar un protocolo de transmisión de vídeo adaptativo:

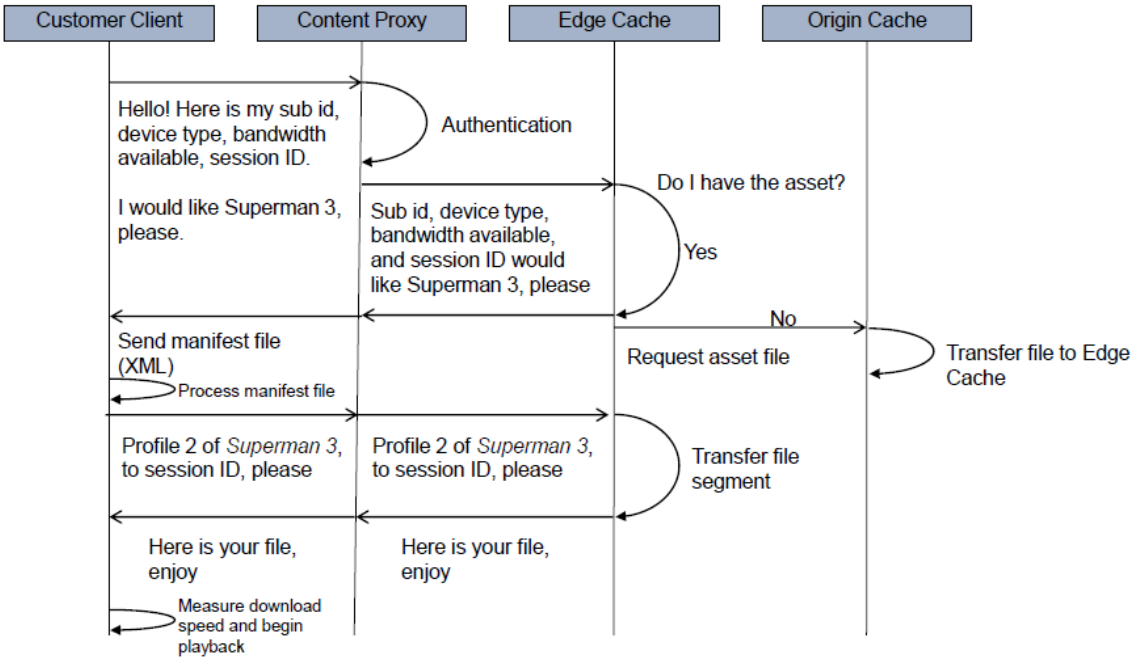


Figura 5. Diagrama de flujo entre cliente y servidor

En primer lugar el cliente se autentica indicando su identificador, el dispositivo que utiliza, el ancho de banda disponible; y solicita el contenido que quiere ver. Se solicita el vídeo al servidor más cercano, en caso de que no lo tuviera en caché se solicita al Servidor Origin. En cualquier caso y siempre que la petición sea correcta, se manda al cliente el fichero XML (Manifest). El cliente puede ver el número de calidades de las que dispone el contenido de vídeo y elegir aquella que más se ajuste a sus características. Responde al servidor con el profile requerido y éste envía el fichero en pequeños fragmentos para que el reproductor del cliente pueda reproducir el vídeo.

Posteriormente pasaremos a estudiar cada uno de los protocolos de transmisión de vídeo adaptativo que nos permiten llevar a cabo la anterior comunicación.

2.3. CONCEPTOS

Antes de entrar de lleno a estudiar cada uno de los protocolos, vamos a definir algunos conceptos que serán básicos para la realización del proyecto.

2.3.1. CDN

Una CDN (Content Delivery Network) es un conjunto de servidores situados en diversos puntos de una red que se encargan de distribuir el contenido de manera eficiente, para ello se lleva a cabo un balanceo de carga entre los servidores y se sirve el contenido desde el servidor más cercano al cliente. Se suelen utilizar para almacenar y distribuir contenidos de un sitio web o para la distribución de vídeo (streaming). Las CDNs suelen ser gestionadas por empresas que se dedican a la distribución de contenido de forma específica.

Una CDN se hace imprescindible cuando se quiere dar servicio a un número de usuarios medio ya que nos ofrece ventajas como:

- Mayor capacidad de conexión.
- Disminución del tiempo de respuesta cara a la entrega de contenido al usuario.
- Disminución de los costes asociados a la distribución del contenido.
- Reducción de la pérdida y demora de paquetes al emplear nodos cercanos al usuario.
- Disminución de la carga en la red.
- Redundancia, ante la caída de un servidor no se pierde el servicio.
- Ofrece estadísticas de los usuarios sobre el consumo de contenidos, localización geográfica, …

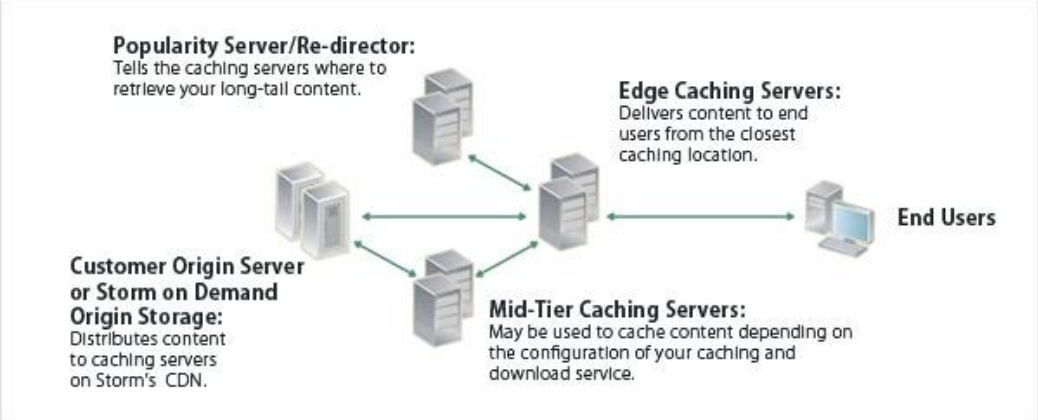


Figura 6. Estructura de una posible CDN

2.3.2. Propiedad

Para la realización de las pruebas de nuestro proyecto utilizaremos lo que se conocen como propiedades.

Una propiedad es un conjunto de reglas que comparten un nombre común y se aplican a todos los streams de audio/vídeo que contenga. Se solicitan al proveedor de servicios de CDN.

Llevan asociada una url pública para poder acceder a ella. En nuestro caso todas las propiedades que emplearemos serán tipo *Caching Live*, es decir, propiedades que utilizan los protocolos de transmisión de vídeo adaptativo que vamos a estudiar.

2.3.3. Métodos de ingesta de contenido en la CDN

Hay dos métodos para insertar contenido en la CDN: Push y Pull.

En el método Pull es la CDN quien va a buscar el contenido a un servidor local, mientras que en el método Push es el cliente quien deposita el contenido en la CDN, es decir, desde los codificadores del cliente se ingesta el contenido en un servidor de la CDN.

2.4. Smooth Streaming (HSS)

Smooth Streaming o HSS es el protocolo de Microsoft para la transmisión de vídeo por Internet.

Esta tecnología fue utilizada por primera vez por Microsoft para distribuir vídeo bajo demanda (*video on demand*) en las Olimpiadas de verano del año 2008 para NBCOlympics.com. Lo que se hizo fue producir múltiples ficheros .wmv (Windows Media Video) con diferentes bit-rates y resolución. Estos ficheros se troceaban en segmentos de duración 2 segundos. Los segmentos se subían a una CDN para posteriormente descargarlos y reproducirlos en un player de Silverlight.

Smooth Streaming funciona bajo HTTP y el cliente principal que utiliza es Silverlight.

El objetivo se centra en proporcionar al usuario la mejor calidad posible de vídeo adecuándose a las características del dispositivo utilizado por el cliente así como a las condiciones de la red en la que se trabaje. Para usuarios que dispongan de dispositivos potentes y de conexiones de gran ancho de banda, se puede distribuir el vídeo en calidad HD. Lo que se persigue es que el usuario pueda recibir una calidad de vídeo óptima acorde con las características tanto de la red como del dispositivo que utilice, y esa calidad va cambiando en función de esas condiciones.

Los protocolos de transmisión de vídeo adaptativo que se estudian en este proyecto basan su funcionamiento en el envío de pequeños fragmentos en formato MPEG-4 (caso de Smooth Streaming) o MPEG2-TS. Según se reproduce un contenido de vídeo, las condiciones de la red pueden variar (reducción de ancho de banda), así como las características de los dispositivos (carga cpu, resolución pantalla). Ante estas circunstancias, el cliente puede solicitar que el siguiente fragmento venga de un stream codificado a una calidad diferente para amoldarse a las nuevas condiciones de trabajo. De esta forma el usuario podrá seguir con la reproducción del contenido sin cortes y con la mejor calidad disponible.

2.4.1. Arquitectura

Smooth Streaming está basado en el formato MPEG-4, en la especificación de ISO/IEC 14496-12 ISO Base Media File Format [3].

La unidad básica de un fichero MP4 se denomina “box”. Cada “box” puede contener tanto datos como metadatos. La organización de los datos y metadatos de un fichero se pueden organizar de dos maneras. En algunos casos es útil que los metadatos aparezcan antes que los datos de tal forma que el reproductor del cliente tenga información de antemano del contenido a reproducir. Pero hay otros casos, por ejemplo en algunos casos de Live Streaming, en los que esto no es posible bien porque los metadatos sean desconocidos o porque nos interese que la cabecera de los ficheros sea lo más corta posible. Un fichero de Smooth Streaming presenta el siguiente aspecto:

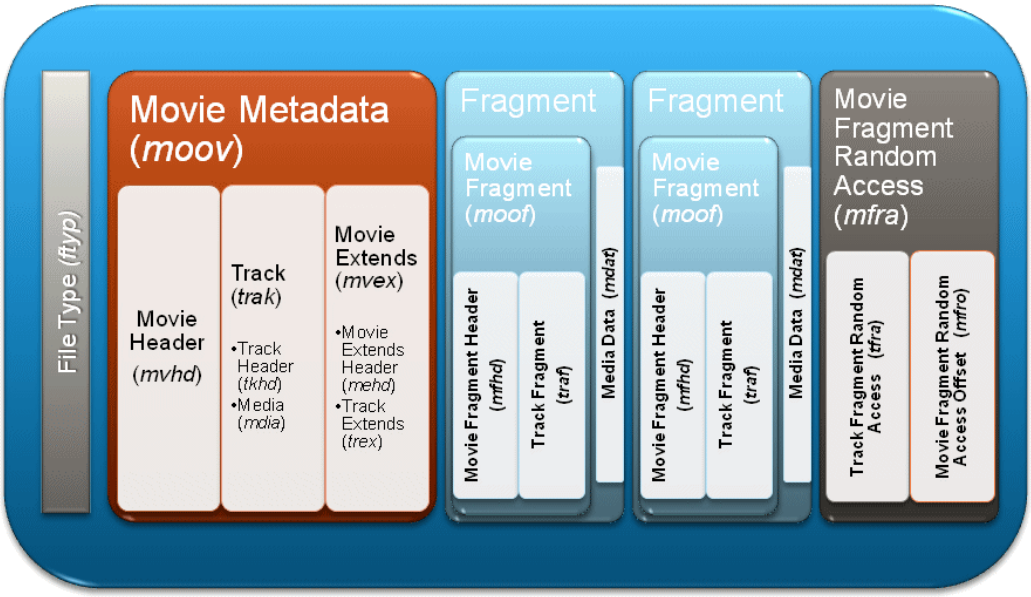


Figura 7. Formato fichero Smooth Streaming

El fichero comienza con una serie de metadatos ('moov') que describen las características del fichero de vídeo, pero es en los propios fragmentos dónde tenemos una serie de metadatos más precisos acerca de cada uno de ellos ('moof'), además de los datos multimedia ('mdat'). La duración de cada fragmento en Smooth Streaming suele ser de 2 segundos. El fichero se cierra con un box denominado 'mfra' que permite una búsqueda precisa y fácil del fichero.

Un fragmento de Smooth Streaming presenta el siguiente aspecto:

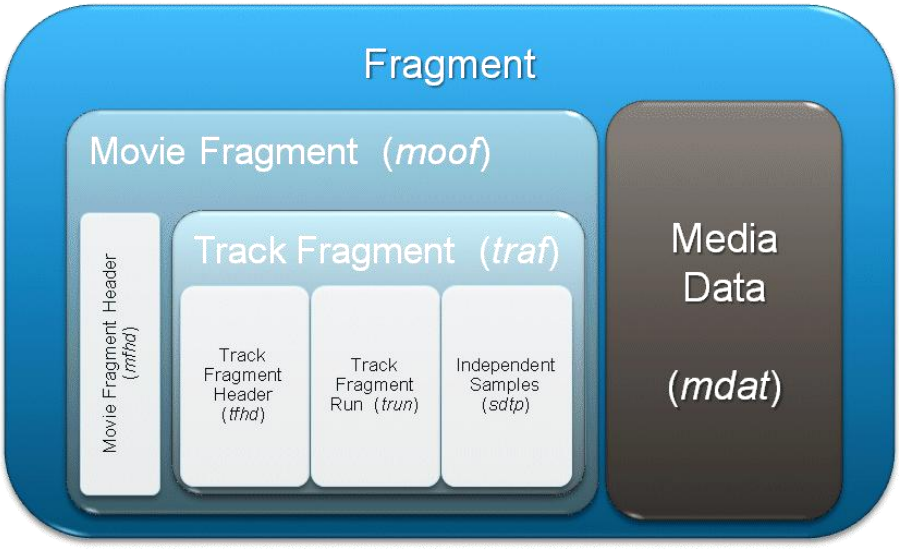


Figura 8. Formato de un fragmento de Smooth Streaming

Un contenido en Smooth Streaming se compone de diferentes ficheros:

- ***.ismv:** contiene vídeo y audio, o sólo vídeo. Hay un fichero *.ismv por calidad.
- ***.isma:** sólo contiene audio. Si el audio va acompañado de vídeo se podría incluir en el fichero *.ismv.
- ***ism:** es el manifest del servidor. Describe la relación entre el contenido multimedia, bit-rates y archivos del servidor. Este fichero sólo es utilizado por el servidor.
- ***.ismc:** es el manifest del cliente. Contiene los streams disponibles para el cliente y el tipo de códec utilizado, así como el bit-rate, la resolución de vídeo, etc… Es el primer archivo que se envía al cliente.

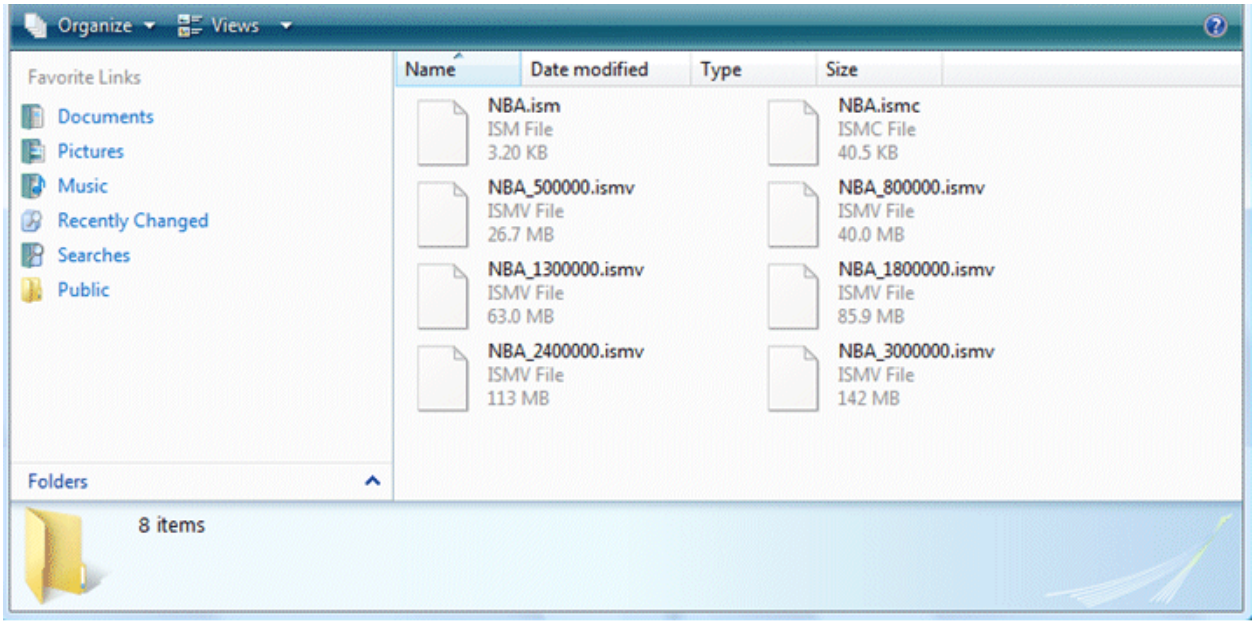


Figura 9. Arquitectura de archivos de Smooth Streaming

2.4.2. Manifest Smooth Streaming

Es un fichero XML.

```
<?xml version="1.0" ?>
<SmoothStreamingMedia MajorVersion="2" MinorVersion="0" Duration="0" TimeScale="10000000" IsLive="TRUE"
  LookAheadFragmentCount="2" DVRWindowLength="600000000" CanSeek="TRUE" CanPause="TRUE">
- <Protection>
    <ProtectionHeader SystemID="9a04f079-9840-4286-ab92-
      e65be0885f95">bAYAAAEAAQBiBjwAVwBSAE0ASABFAEEARABFAFIAIAB4AG0AbABuAHMAPQAiAGgAdAB0AHA
  </Protection>
- <StreamIndex Type="video" Name="video" Language="und" Subtype="" Chunks="0" TimeScale="10000000"
    Url="QualityLevels({bitrate})/Fragments(video={start time})">
    <QualityLevel Index="0" Bitrate="2500000"
      CodecPrivateData="00000016764001fac1b2940b033fbc052828282a00000300200000065c0c0004c4b0000e
      FourCC="AVC1" MaxWidth="704" MaxHeight="396" />
    <QualityLevel Index="1" Bitrate="300000"
      CodecPrivateData="00000001674d401f8d94a0a0cfcf80a50505054000000300400000cb80000493c0006dda7
      FourCC="AVC1" MaxWidth="320" MaxHeight="180" />
    <QualityLevel Index="2" Bitrate="1400000"
      CodecPrivateData="00000001674d401f8d94a05017fcb80a50505054000003000400000300cb81800155cc000
      FourCC="AVC1" MaxWidth="640" MaxHeight="360" />
    <QualityLevel Index="3" Bitrate="700000"
      CodecPrivateData="00000001674d401f8d94a04012d80a50505054000003000400000300cb810002ab900010
      FourCC="AVC1" MaxWidth="512" MaxHeight="288" />
    <c d="20000000" t="999207835277772" />
    <c d="20000000" />
    <c d="20000000" />
```

Figura 10. Archivo XML correspondiente al Manifest de Smooth Streaming

Si accedemos al Manifest nos encontramos la información de la figura de arriba. En primer lugar hay una serie de metadatos relativos a las características del contenido a distribuir. Podemos ver que el contenido es "live" o el valor de la etiqueta *DVRWindowLength*, entre otros parámetros. El DVR (Digital Video Recorder) es una función de los reproductores de vídeo para contenido *live* que permite dar marcha atrás aunque el contenido siga avanzando o volver a ver una parte que se haya perdido.

También se puede ver que el tipo de contenido es vídeo y a continuación se muestran las distintas calidades de vídeo que el cliente puede elegir. En el caso anterior tenemos hasta 4 calidades, cada una con un determinado bit-rate y una resolución específica. También se puede ver si la propiedad está encriptada o no (información de la etiqueta <*Protection*>) y la duración que tiene cada fragmento (en este caso 2 segundos).

2.4.3. Distribución del contenido en Smooth Streaming

Cuando un cliente quiere reproducir un contenido mediante este protocolo, lo primero que hace es solicitar al servidor el archivo ***.ismc.** Una vez recibido el Manifest el cliente puede ver los bit-rates y resoluciones disponibles, así como una lista de los chunks disponibles (así como la duración de cada uno). El cliente elige una calidad y solicita los fragmentos. Cuando el servidor recibe la petición del cliente, busca la calidad en el archivo *.ism y lo mapea con el fichero *.ismv o *.isma asociado. Posteriormente lee el archivo en formato MP4 y fijándose en la 'tfra' box (ver figura 7), averigua que fragmento ('moof'+'mdat') tiene que enviar primero. Se extrae y se envía al cliente. El fragmento se queda en caché para que en el caso de que otro cliente lo solicite no se tenga que pedir al servidor origen.

Una de las características de Smooth Streaming es que el Manifest (también conocido como Playlist) sólo se pide una vez. Cada fragmento lleva en la cabecera información relativa a los siguientes. Esto también ocurre en HDS pero no en HLS.

2.4.4. Dispositivos de Smooth Streaming

Smooth Streaming es la solución de Microsoft y principalmente va enfocado a dispositivos tales como:

- PC’s
- TV Conectadas
- Xbox
- Windows Phone.

Si quisiéramos utilizar Smooth Streaming en un iPad no va a funcionar.

2.5. HLS

HLS (HTTP Live Streaming) es el protocolo de Apple para la transmisión de vídeo por Internet.

Al igual que en Smooth Streaming el contenido se trocea en pequeños ficheros (denominados chunks) que son descargados por el cliente mediante el protocolo HTTP. Se utiliza tanto para contenido "live" como contenido "On Demand". El stream se puede codificar con diferentes bit-rates lo que nos permite tener diferentes calidades. Además, proporciona encriptación y autenticación sobre HTTPS.

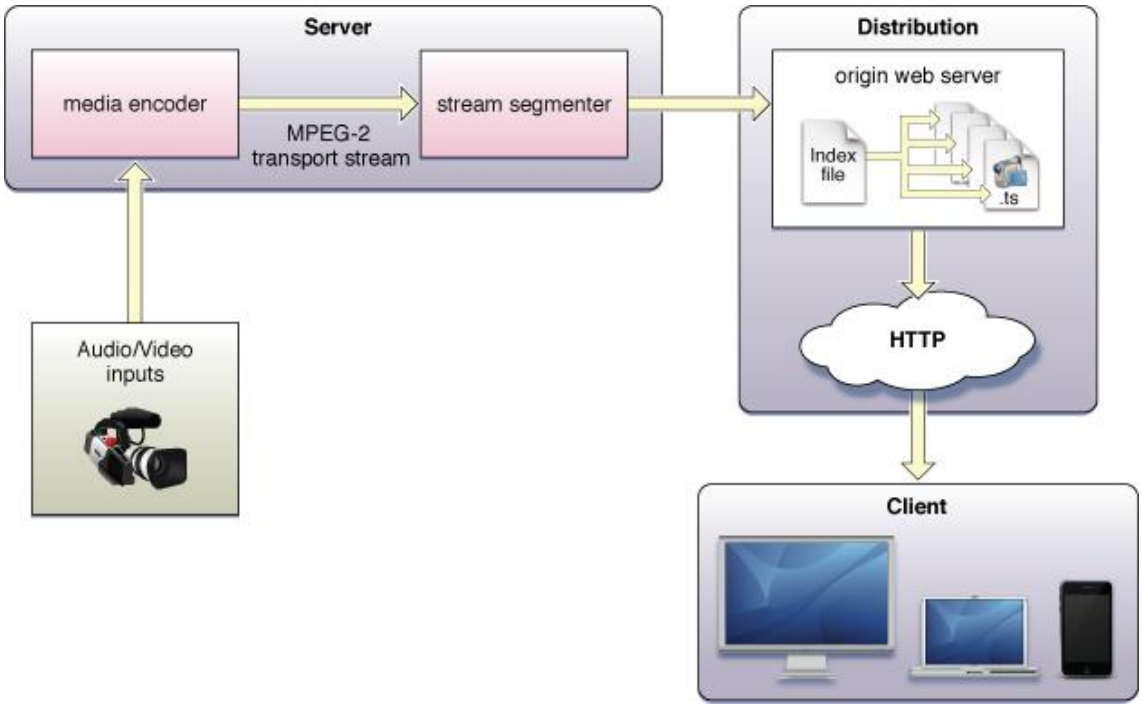


Figura 11. Protocolo de transmisión de vídeo HLS

En el caso de HLS, el Index file (playlist) se refresca de forma periódica (recordamos que en Smooth Streaming sólo se pide una vez).

Como se puede ver en la figura 11, HLS está caracterizado por 3 elementos básicos: la parte del servidor, la parte de distribución y la parte del cliente.

La parte del servidor se encarga de coger los streams de entrada y codificarlos, normalmente el vídeo se codifica como H.264 y el audio como AAC (Advanced Audio Encoding), conformando un MPEG-2 Transport Stream [4] que se trocea en segmentos (archivos .ts) para posteriormente poder distribuirlos por la red.

En concreto el servidor consta de un codificador y un segmentador.

El codificador (*media encoder*) coge una señal en tiempo real de vídeo-audio, la codifica a una determinada tasa y la encapsula para su transporte. La codificación debe

ser soportada por el dispositivo del cliente (H.264 para vídeo y HE-AAC (High Efficiency-AAC) para audio). El codificador distribuye el contenido en formato MPEG-2 hacia el segmentador (*stream segmenter*). Éste consiste en algún software que lee el Transport Stream y lo divide en pequeños segmentos de igual duración. Además genera el Index file que contiene referencias a los Sub-Manifest que contienen los ficheros .ts.

La parte de distribución está formada por servidores web que es dónde se alojan los segmentos. Estos servidores que forman la CDN se encargan de aceptar las peticiones de los clientes y de la distribución del contenido hacia el cliente. Dentro de estos servidores se encuentran los denominados *mid-tier o servidores intermedios*, pertenecen a la CDN y es dónde se encuentra el contenido cacheado. Estos servidores permiten que el número de consultas a los servidores Origin (origen) disminuya, ya que cuando un servidor de frontera (denominado Edge-Server) no tiene el contenido solicitado por el cliente, éste acude al *mid-tier* en lugar de al origen.

Por último tenemos la parte del cliente que es la que se encarga de elegir la calidad del contenido a descargar, descargarla y reproducirla sin cortes.

2.5.1. Arquitectura

La arquitectura de archivos de HLS está compuesta por:

El archivo Manifest es tipo **.m3u8** y es el primer archivo al que se accede cuando queremos reproducir un stream de HLS. En él aparecen las calidades disponibles. Los ficheros **.m3u8** son estáticos.

El Sub-Manifest es un archivo en el que se muestran, según la calidad, la secuencia de archivos de vídeo **.ts** que se debe descargar el cliente para la reproducción del contenido. El número de los archivos **.ts** es constante pero éstos se van actualizando de forma secuencial, es decir, si se pide el Sub-Manifest pasado un tiempo la secuencia de números **.ts** se ve incrementada respecto a un Sub-Manifest anterior.

Esos archivos **.ts** son conocidos como chunks y contienen el vídeo. La duración típica de estos archivos suele ser de 10 segundos.

2.5.2. Manifest HLS

Es un fichero .m3u8

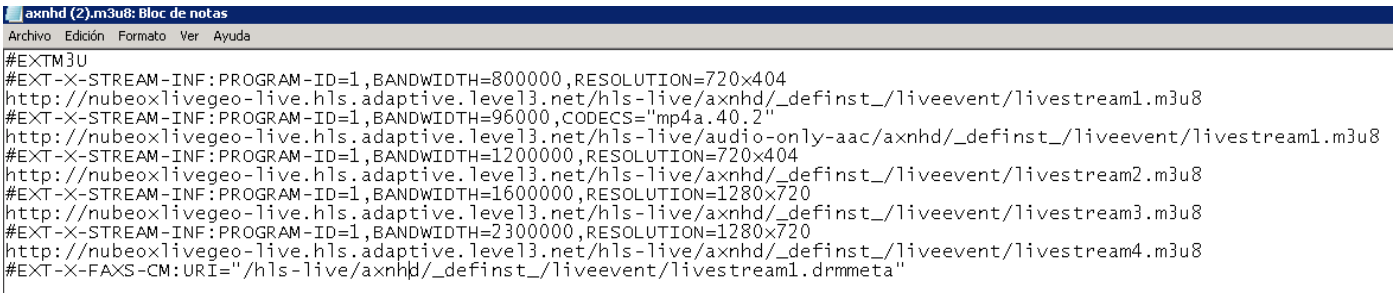


Figura 12. Manifest HLS

En el Manifest se indica la URL pública para acceder y coger una determinada calidad. En el caso del Manifest anterior tenemos un total de 5 calidades, identificadas como "livestreami" siendo i=1,…4. Además en la última línea hay información adicional para obtener información de la parte de DRM (para la protección de los datos). Si accedemos al Sub-Manifest de cada calidad nos encontraremos con lo siguiente:

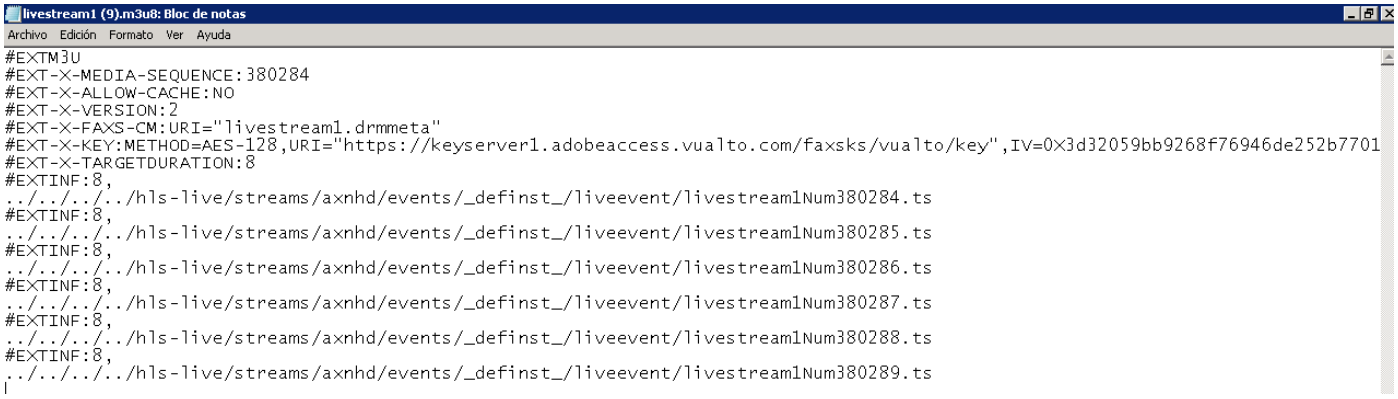


Figura 13. Sub-Manifest correspondiente a una de las calidades HLS

Aquí nos encontramos los chunks (archivo .ts de vídeo) relativos al stream correspondiente a una calidad determinada. Los archivos .ts están ordenados, y como hemos comentado anteriormente, si esperamos un tiempo y volvemos a solicitar el Sub-Manifest veremos como la secuencia de números ha progresado.

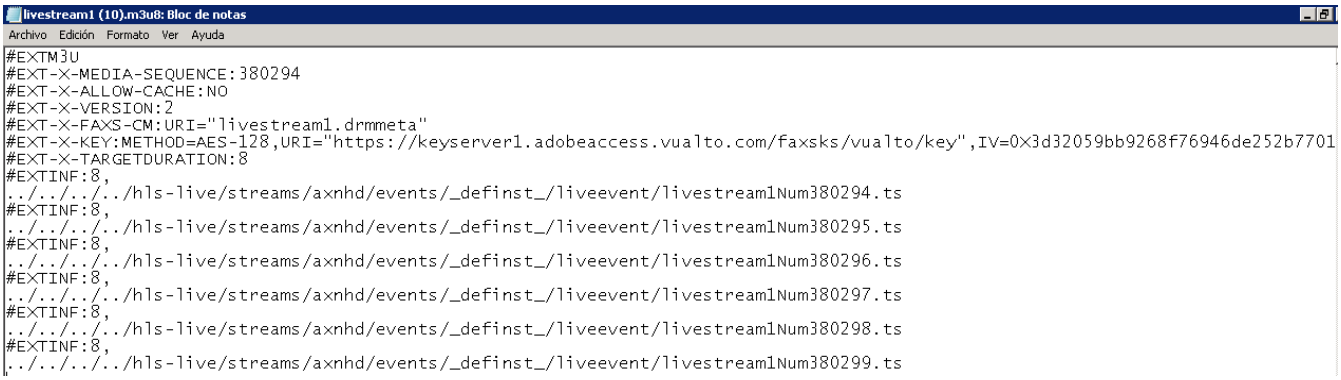


Figura 14. Sub-Manifest del mismo contenido pasado un tiempo

Podemos comprobar que efectivamente se ha incrementado la secuencia de números, en este caso, de un contenido del canal axnhd.

2.5.3. Factores calidades Stream

El principal objetivo que se persigue al disponer de distintas calidades de un mismo contenido es ofrecer al cliente la mayor calidad posible. A la hora de generar una playlist las calidades se fijan dependiendo de varios factores.

Uno de ellos es la resolución del dispositivo que se utiliza. Un dispositivo puede soportar diferentes resoluciones, por lo que en la playlist se pueden indicar diferentes calidades en función de este parámetro que va ligado con el ancho de banda:

#EXT-X-STREAM-INF:BANDWIDTH=1280000,RESOLUTION=640x360

#EXT-X-STREAM-INF:BANDWIDTH=1700000,RESOLUTION=1280x720

#EXT-X-STREAM-INF:BANDWIDTH=3500000,RESOLUTION=1920x1080

Por ejemplo, si utilizamos un modelo de iPhone antiguo (3GS) el cliente elegirá la calidad con resolución 640x360 ya que el dispositivo no soporta el resto de calidades que se ofrecen en el fichero Manifest. Sin embargo, si utilizamos un iPad que permite el uso de mayores resoluciones, el cliente elegirá la resolución que le dé mejores prestaciones.

Si por un casual estamos utilizando un dispositivo que admite varias resoluciones, pero utilizamos una aplicación con una pantalla pequeña (640x360), se elegirá la calidad respectiva asociada. Si en algún momento de la reproducción se cambiara a pantalla completa, se realizará un cambio (adaptativo) a una calidad superior siempre y cuando sea posible (lo permitan las condiciones de la red o cualquier otro factor influyente).

Otro aspecto sobre el que se podría basar la distinción de calidades podría ser el formato de codificación de vídeo, y hacer que el cliente elija una calidad u otra en función del perfil de codificación utilizado. Por ejemplo, podríamos tener una primera calidad para un perfil *H.264 Baseline* [6] [7], una segunda para un perfil *H.264 Main* y una tercera

para un perfil *H.264 High*. Dependiendo de si un dispositivo soporta un tipo de codificación u otro, el cliente elegirá la calidad que mejor se adecúe al mismo.

O simplemente las calidades podrían basarse en el modelo del dispositivo a utilizar (iPhone frente a un iPad). Para ello en la cabecera de la petición HTTP se añade un campo denominado *User-Agent.*

Aparte de tener en cuenta las características de los dispositivos que utilizamos, el estado y la capacidad de la red también son muy importantes. No vamos a tener las mismas condiciones si estamos conectados a una red de datos (3G) o a una red Wifi.

Normalmente se recomienda que la primera calidad que aparece en la playlist tenga asociada un bit-rate tal que la gran mayoría de usuarios pueda soportar. La reproducción del stream suele comenzar con la primera calidad y es con la que se determina si es preciso realizar algún cambio o no. El orden del resto de calidades es irrelevante.

Se recomienda que los bit-rates de dos calidades contiguas difieran entre sí en un factor 1.5 o 2, de tal forma, que no se malgaste el ancho de banda en el caso de que fueran demasiado próximos y de que el cambio sea viable, cosa que podría no ocurrir si los bit-rates estuvieran muy alejados entre sí.

A continuación podemos ver una recomendación de las condiciones de codificación (resolución, bit-rate, tipo de codificación de vídeo) para streams tipo HLS diferenciando si utilizamos una red de datos (celular) o una red Wifi.

16:9 Aspect Ratio

	Dimensions	Frame Rate *	Total Bit Rate	Video Bit Rate	Audio Bit Rate**	Audio Sample Rate	Keyframe***	Restrict Profile to:
CELL	416x234	12	264	200	64	48	36	Baseline, 3.0
CELL	480x270	15	464	400	64	48	45	Baseline, 3.0
WiFi/Cell	640x360	29.97	664	600	64	48	90	Baseline, 3.0
WiFi	640x360	29.97	1296	1200	96	48	90	Baseline, 3.1
WiFi	960x540	29.97	3596	3500	96	48	90	Main, 3.1
WiFi	1280x720	29.97	5128	5000	128	48	90	Main, 3.1
WiFi	1280x720	29.97	6628	6500	128	48	90	Main, 3.1
WiFi	1920x1080	29.97	8628	8500	128	48	90	High, 4.0

Tabla 3. Condiciones de codificación HLS para 16:9

4:3 Aspect Ratio								
	Dimensions	Frame Rate *	Total Bit Rate	Video Bit Rate	Audio Bit Rate**	Audio Sample Rate	Keyframe***	Restrict Profile to:
CELL	400x300	12	264	200	64	48	36	Baseline, 3.0
CELL	480x360	15	464	400	64	48	45	Baseline, 3.0
WiFi/Cell	640x480	29.97	664	600	64	48	90	Baseline, 3.0
WiFi	640x480	29.97	1296	1200	96	48	90	Baseline, 3.1
WiFi	960x720	29.97	3596	3500	96	48	90	Main, 3.1
WiFi	1280x960	29.97	5128	5000	128	48	90	Main, 3.1
WiFi	1280x960	29.97	6628	6500	128	48	90	Main, 3.1
WiFi	1920x1440	29.97	8628	8500	128	48	90	High, 4.0

Tabla 4. Condiciones de codificación HLS para 4:3

2.5.4. Formatos de vídeo y audio soportados en HLS

La implementación que ofrece Apple de HLS soporta una serie de formatos de vídeo y audio.

- Vídeo:
    - H.264 Baseline Level 3.0, Baseline Level 3.1, Main Level 3.1, and High Profile Level 4.1 [6] [7].
- Audio:
    - HE-AAC o AAC-LC hasta 48 kHz, audio estéreo.
    - MP3 (MPEG-1 Audio Layer 3) 8 kHz a 48 kHz, audio estéreo.
    - AC-3 (para Apple TV, solo en modo “pass-through”).

El formato de vídeo depende del dispositivo a utilizar. Por ejemplo, el formato H.264 Baseline 3.0 funciona en todos los dispositivos; el H.264 Baseline 3.1 se utiliza para iPhone 3G y posteriores, iPod touch 2nd generation y versiones posteriores; el H.264 Main Profile 3.1 para iPad (todas las versiones), Apple TV 2 y posteriores e iPhone 4 y posteriores; el H.264 Main Profile 4.0 y el H.264 High Profile 4.0 para Apple TV 3 y posteriores, iPad 2 y posteriores e iPhone 4S y posteriores; y el H.264 High Profile 4.1 para iPad 2 y posteriores e iPhone 4S y posteriores.

2.5.5. Dispositivos de HLS

HLS es la solución de Apple y va enfocado a dispositivos tales como:

- iPad
- iPhone
- iPod touch

- Mac
- Apple TV
- Televisiones conectadas

Todos los dispositivos que incluyen la versión de IOS 3.0 y posteriores incluyen software para reproducir contenido HLS. El navegador Safari permite la reproducción del contenido HLS a través de una página web en un iPad o en MAC's, además permite pantalla completa para reproducir streams de HLS en dispositivos con pequeña pantalla como iPhone o iPod touch.

2.6. HDS

HDS (HTTP Dynamic Streaming) es la solución de Adobe para la transmisión de vídeo por Internet.

HDS presenta una novedad respecto a los dos protocolos anteriores ya que introduce un nuevo concepto, define una unidad de contenido más pequeña que denomina fragmento. Un fragmento es una unidad descargable que contiene un intervalo de vídeo o audio (de unos 4 segundos según el estándar [16]). Se identifican por un número y se pueden agrupar en segmentos. Los números de los fragmentos, al igual que ocurre en HLS, se van incrementando progresivamente según va avanzando el contenido del stream.

Los fragmentos de HDS utilizan un formato F4V [18] basado en el formato MPEG-4 de la especificación ISO/IEC 14496-12 ISO Base Media File Format [3], como ocurría para el caso de Smooth Streaming.

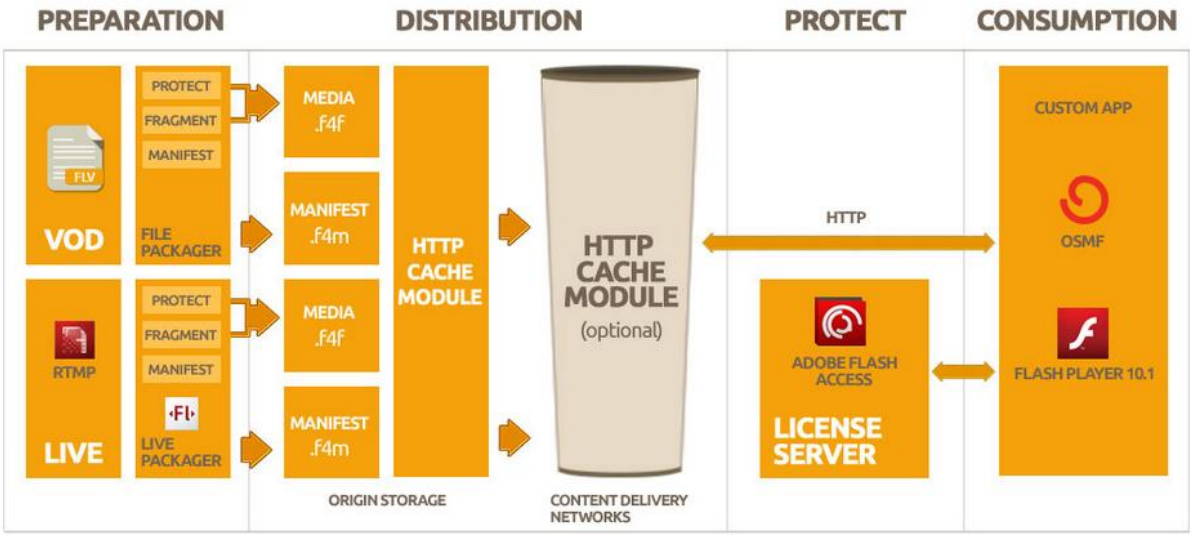


Figura 15. Proceso de preparación y distribución de HDS

La unidad de contenido más grande es el segmento cuya función es compactar los fragmentos para fomentar la eficiencia en los servidores.

Los streams pueden presentar contenido “live” o “video on demand”, y se dispondrá de diferentes calidades para que el cliente pueda elegir la que más se ajuste a sus características.

En HDS se manejan dos tablas para poder identificar los segmentos o fragmentos:

- Fragment Run Table: asocia los intervalos del stream con los números de los fragmentos.
- Segment Run Table: asocia los números de los fragmentos con los números de los segmentos. Esta información suele ir en el archivo Manifest.

2.6.1. Manifest HDS

Es un documento XML en formato .f4m. Contiene información para poder procesar un stream HDS. Lleva información de los segmentos y fragmentos. En el Manifest se especifica el tipo de contenido (audio, vídeo,…), el bit-rate y, opcionalmente, información de seguridad.

```
<manifest xmlns="http://ns.adobe.com/f4m/2.0">
  <baseURL>http://nubeoxlivegeo-live.hds.adaptive.level3.net/hds-live/axnhd/_definst_/liveevent/</baseURL>
  <dvrInfo windowDuration="60"/>
  <media href="livestream1.f4m" bitrate="800"/>
  <media href="livestream2.f4m" bitrate="1200"/>
  <media href="livestream3.f4m" bitrate="1600"/>
  <media href="livestream4.f4m" bitrate="2300"/>
</manifest>
```

Figura 16. Manifest HDS

En el Manifest nos encontramos con la url pública de acceso e información del DVR (Digital Video Recorder) que a través de la etiqueta <dvrInfo> nos indica el tiempo durante el cual se puede grabar el contenido que se reproduce en directo.

Además, se muestran las diferentes calidades en las que se ofrece el contenido. Son versiones de un mismo audio o vídeo con diferente bit-rate. En el caso del ejemplo anterior tenemos un total de 4 calidades. Si accedemos al archivo .f4m de una de ellas obtendríamos:

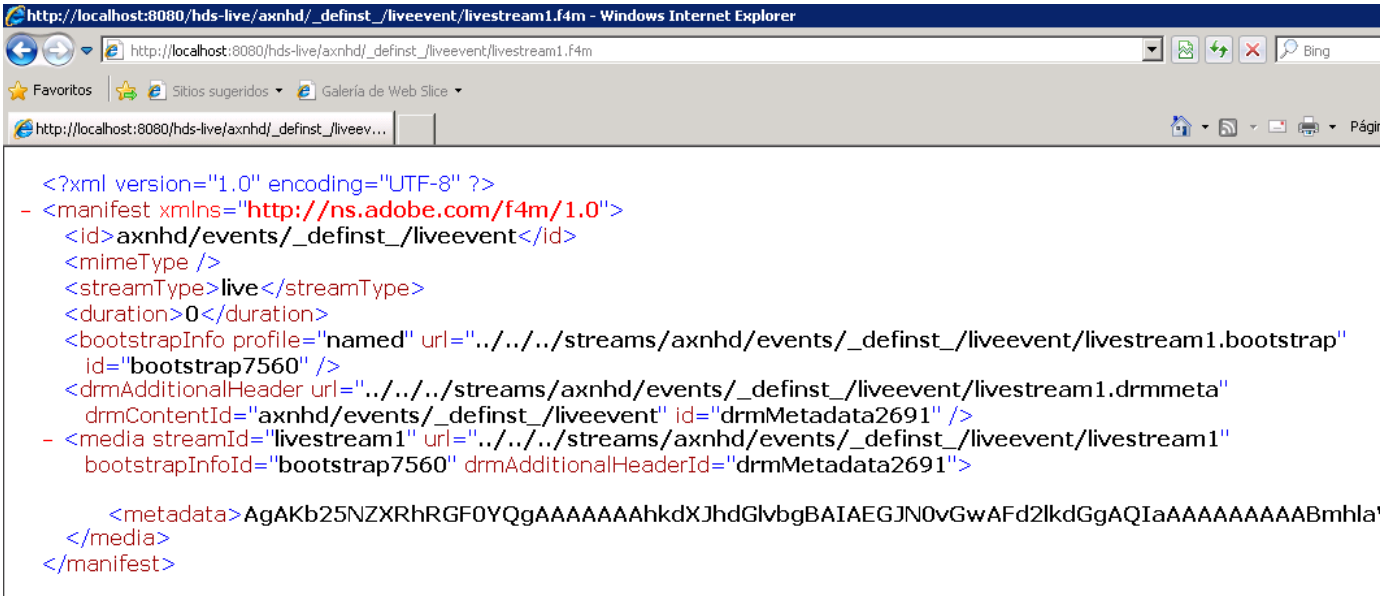


Figura 17. Fichero .f4m de una de las calidades del stream HDS

Dentro de este fichero tenemos la información de "*bootstrap*" o información de arranque o inicialización. Esta información va codificada en un fichero binario que no se puede abrir con cualquier programa habitual. El "bootstrap" indica la secuenciación y proporciona un mapeado entre los números de fragmento y el intervalo de tiempo del stream, así como de la relación entre los números de segmentos y los fragmentos (tablas que definimos anteriormente).

Resumiendo, por un lado tenemos la información de las distintas calidades para el stream HDS, en cuanto al vídeo o las opciones de audio; y por otro lado dentro de cada calidad, tenemos la información de bootstrap y la de seguridad (DRM).

2.6.2. Funcionamiento

Un cliente puede solicitar streams *On Demand* o *Live* streams (al igual que en el resto de protocolos).

En caso de que se solicite un contenido On Demand, el cliente solicita el Manifest que viene especificado por una url pública (en HDS sólo se pide la playlist una vez). El servidor devuelve este archivo. Como hemos visto anteriormente, el Manifest contiene el número de calidades en las que se puede ofrecer el contenido y puede contener también las opciones de lenguaje. Atendiendo a las necesidades del cliente (bit-rate, idioma) se pide el sub-manifest correspondiente. Para VOD streams se realiza una única adquisición del Manifest al comienzo de la reproducción, ésta comienza con el primer fragmento que se indica en la información de bootstrap. Los fragmentos se van descargando y guardando en un buffer. Cada fragmento es independiente por lo que se podrá decodificar de forma independiente. Esto también permite al usuario comenzar la reproducción en un punto aleatorio.

Para streams de tipo Live se sigue el mismo proceso que para VOD streams. El Manifest se solicita una única vez al inicio de la reproducción. Inicialmente, el cliente descarga una secuencia de fragmentos cercanos al punto live de reproducción del stream, éstos se indican en la información de bootstrap. El número de fragmentos descargados inicialmente debe ser tal que se llene el buffer del cliente. Una vez iniciada la reproducción, el cliente continúa descargando fragmentos. El cliente debe "refrescar" el sub-manifest una vez que haya recibido todos los fragmentos que se indican en él.

En el archivo manifest se muestran todas las calidades en las que se puede reproducir el contenido. El cliente debe elegir una de ellas para comenzar la reproducción en función del dispositivo que utilice y las condiciones de la red. En función de esas condiciones, el bit-rate puede variar y eso nos lo permiten todos estos protocolos de transmisión de vídeo adaptativo de los que estamos hablando.

2.6.3. Tamaño del buffer

El tamaño del buffer debe ser tal que se produzcan las mínimas interrupciones en una reproducción teniendo en cuenta factores como las condiciones de la red, el retardo deseado o la escalabilidad del servidor. Se recomienda que el tamaño del buffer sea al menos tres veces la duración de un fragmento.

2.6.4. Comportamiento del servidor

El servidor recibe peticiones HTTP devolviendo un Manifest válido para la petición del cliente. Si el Manifest solicitado fuera desconocido para el servidor, se responde con una respuesta 400 (Error). El servidor establece la cabecera Content-Type del Manifest

como “application/f4m” (formato HDS). El servidor se encarga de controlar el tiempo de vida del Manifest en caché. Para contenido VOD, el Manifest se declara permanentemente cacheable. Para contenido Live el Manifest se suele tener en caché por más de 1 segundo pero menos de la mitad de la máxima duración del fragmento en un stream de este tipo.

El servidor responde una petición HTTP de un fragmento con el contenido de ese fragmento si lo tiene, en caso de no reconocer el fragmento solicitado, se envía una respuesta de error 400. Si el servidor reconoce el fragmento de la petición pero aún no lo tiene disponible, responderá con una respuesta HTTP 503, indicando al cliente que vuelva a solicitar el fragmento pasado un tiempo.

2.6.5. Formatos de vídeo y audios soportados en HDS

En la especificación del formato F4V [18] que emplea HDS se indican los formatos de vídeo y audio que soporta el protocolo:

- Vídeo:
    - GIF.
    - PNG.
    - JPEG.
    - H.264 (Baseline, High, Main, Extended).
    - VP6.
- Audio:
    - MP3.
    - AAC (main profile, low complexity, HE/SBR).

2.6.6. Dispositivos de HDS

HDS se puede utilizar en cualquier dispositivo compatible con Adobe Flash. Principalmente se utiliza en ordenadores (sobremesa o portátiles). Este protocolo no está soportado en dispositivos iOS (no admiten reproductor Flash) por lo que su uso en dispositivos móviles está algo limitado.

2.7. *MPEG-DASH*

Los tres protocolos presentados anteriormente, Smooth Streaming, HDS y HLS, permiten la distribución de vídeo por Internet de forma adaptativa, pero cada uno exige un encapsulamiento diferente.

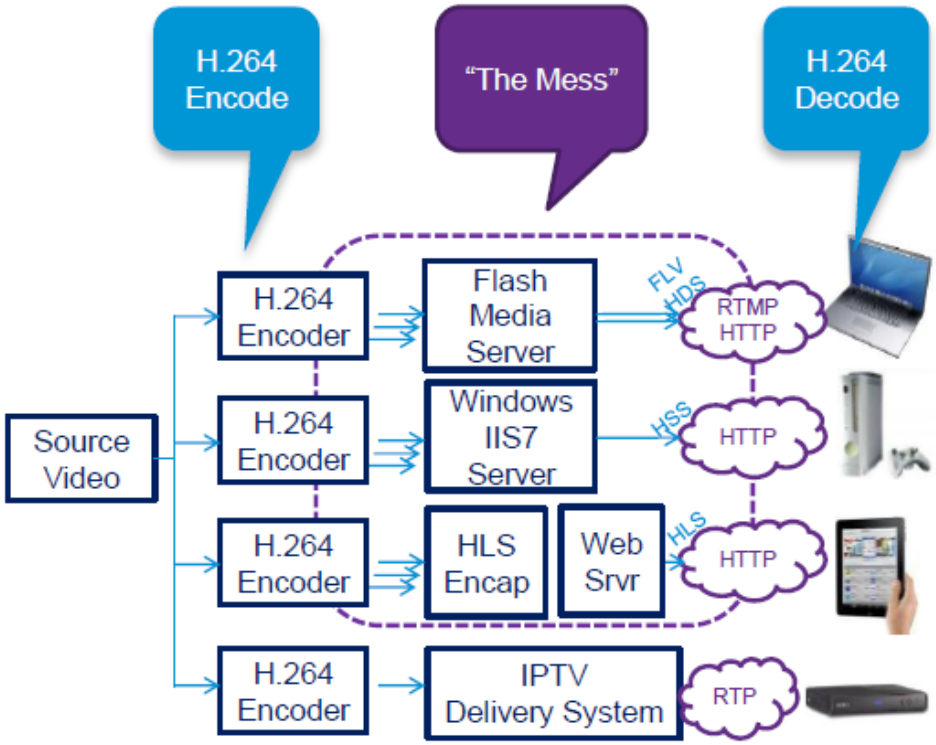


Figura 18. Encapsulamiento de los distintos protocolos estudiados

Para tratar de evitar incompatibilidades entre los distintos protocolos y los dispositivos que utilizan los usuarios, surge MPEG-DASH (Dynamic Adaptive Streaming over HTTP).

La idea es tratar de converger y simplificar la distribución de vídeo por Internet de forma adaptativa, reutilizando la tecnología ya existente.

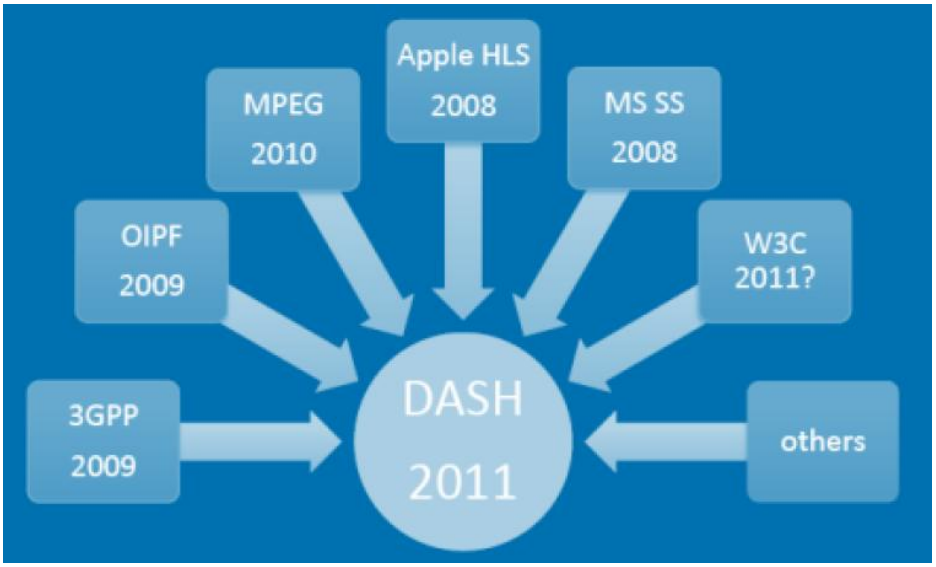


Figura 19. Convergencia de MPEG-DASH

MPEG-DASH se crea con la idea de soportar la distribución del contenido de vídeo basando el control de forma exclusiva en el cliente, que pueda elegir el tipo de codificación de vídeo, el tipo de codificación de audio, el sistema de encriptado o protección de datos,…

2.7.1. Características

Algunas características que definen MPEG-DASH son:

- Soporta la distribución tanto de contenido *On Demand* como de contenido *Live* en formato MPEG-4 y MPEG2-TS.
- Utiliza HTTP. Reutilización y uso eficiente de los recursos: CDN's, proxies, cachés, firewalls.
- Control total de la sesión del streaming por parte del cliente.
- Soporta cambio de calidades sin cortes.
- Admite segmentos con duraciones variables.
- Soporta múltiples sistemas de DRM (protección del contenido).
- El contenido puede estar disponible en múltiples url's (diferentes servidores) aportando redundancia y permitiendo que el usuario pueda maximizar el ancho de banda de la red.

2.7.2. Arquitectura

MPEG-DASH estandariza la información del contenido para asegurar la interoperabilidad entre servidores y clientes. Para ello realiza la definición de varios conceptos: periodos, segmentos, representaciones y sets de adaptación.

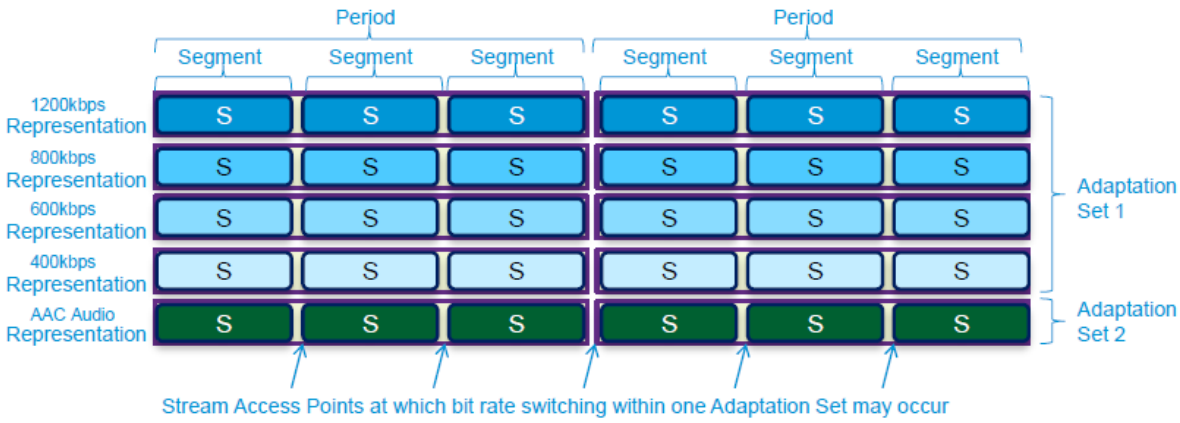


Figura 20. Arquitectura de MPEG-DASH

2.7.2.1. Periodos

Es la unidad más grande que se define en MPEG-DASH. El número de periodos depende de la duración completa del stream. Los periodos aparecen ordenados por el tiempo, que se va incrementando progresivamente.

Los principales atributos son @start y @duration [25].

2.7.2.2. Adaptation Sets

Cada periodo está compuesto por uno o más sets de adaptación. En cada uno de ellos podemos encontrar una o más calidades (también denominadas representaciones).

Las representaciones dentro de un mismo *Adaptation Set* son distintas calidades de una misma fuente (mismo contenido), es decir, diferentes versiones codificadas a diferentes bit-rates, o con un idioma diferente, o con resolución diferente, etc.

Algunos de los principales atributos son: @group, @lang, @mediaComponentTytpe, @par, @minBandwidth, @maxBandwidth, @minWidth, @maxWidth, @minHeight, @maxHeight, @minFrameRate, @maxFrameRate, @segmentAlignment, @bitStreamSwitching, @subsegmentAlignment [25].

2.7.2.3. Representations

Son las distintas calidades que el cliente puede elegir de un mismo contenido. Pueden diferir en algunos parámetros de codificación tales como: bit-rate, resolución, lenguaje, códec,…

Las representaciones están alineadas con las líneas de tiempo que marcan los periodos. Consisten de uno o más segmentos. Pueden contener un segmento de inicialización si no están ya inicializadas. También pueden contener alguna sub-representación o bien no presentar ninguna. La sub-representación lo que hace es permitir el acceso a una versión de menor calidad de la representación elegida.

2.7.2.4. Segmentos

Un segmento es una unidad que lleva información de la url y la ventana de tiempo que ocupa en el stream y en la que puede ser accedido a través de dicha url. En un segmento pueden aparecer varias url's lo que nos permite tener el contenido en diferentes localizaciones proporcionando redundancia.

Existe un segmento de inicialización para cada representación que permite iniciar la reproducción del contenido (archivo .m4i)

Se utilizan dos formatos de segmentos básicos:

- Por un lado basados en ISO Base Media File Format definido en la norma ISO/IEC 14496-12 [3].
- Por otro lado basado en el formato de MPEG-2 Transport Stream definido en la norma ISO/IEC 13818-2 [5].

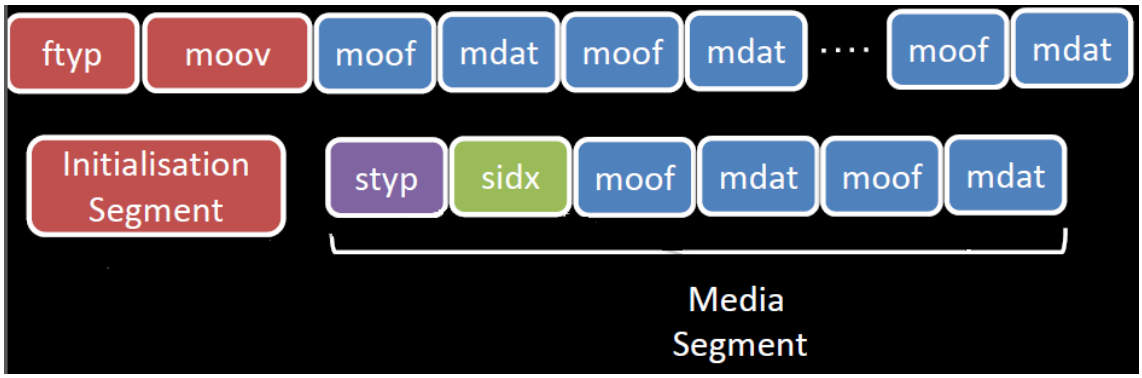


Figura 21. Formato segmento MPEG-DASH ISO Base Media File Format

Un segmento de MPEG-DASH tiene una estructura compuesta por cajas ("box"). Los segmentos pueden ser cortos o largos. El segmento de inicialización viene definido por las cajas 'ftyp' y 'moov'. La otra parte del segmento se denomina "*Media Segment*". En las cajas 'moof' y 'mdat' está el propio contenido del stream. La caja 'styp' es opcional y se utiliza para el etiquetado de archivos. La caja 'tfad' es para el acceso aleatorio aunque en el ejemplo anterior no se utiliza.

Segment duration	Advantages	Disadvantages
Short	• Suitable for live • for VoD commonality with live • High switching granularity on segment level	• Large number of files • Large number of URLs • Fixed request size • switching granularity on segment level
Long	• Small number of files • Small number of URLs • High switching granularity • Flexible request sizes • Improved cache performance	• Need for Segment Index • Difference from Live

Figura 22. Ventajas y desventajas según la duración del segmento MPEG-DASH

Los segmentos cortos, de duración (1-10 segundos), son apropiados para streams tipo live. Como es lógico se tienen mayor número de segmentos y mayor número de url's.

Los segmentos largos (10 segundos- 2 horas) son menos comunes ya que no son apropiados para contenido "*live*".

2.7.3. Fichero MPD

La información del contenido va en un fichero XML denominado MPD ("Media Presentation Description"). En este fichero se describen los segmentos y su duración.

En este fichero se indica información de los streams que nos permita seleccionar o rechazar un tipo de calidad/representación. Algunos de esos datos podrían ser el tipo de codificación, el idioma, la resolución, el bit-rate,…

Además nos encontraremos:

- La url pública y el rango de bytes de cada segmento accesible.
- La próxima actualización del fichero MPD en el servidor.
- El tiempo de comienzo y fin del segmento.
- El tiempo aproximado de comienzo del stream y la duración del segmento en la línea de tiempo del stream.
- Para servicios tipo *live*, instrucciones para el comienzo de la reproducción de tal forma que no se altere la misma durante la duración total de dicha reproducción.

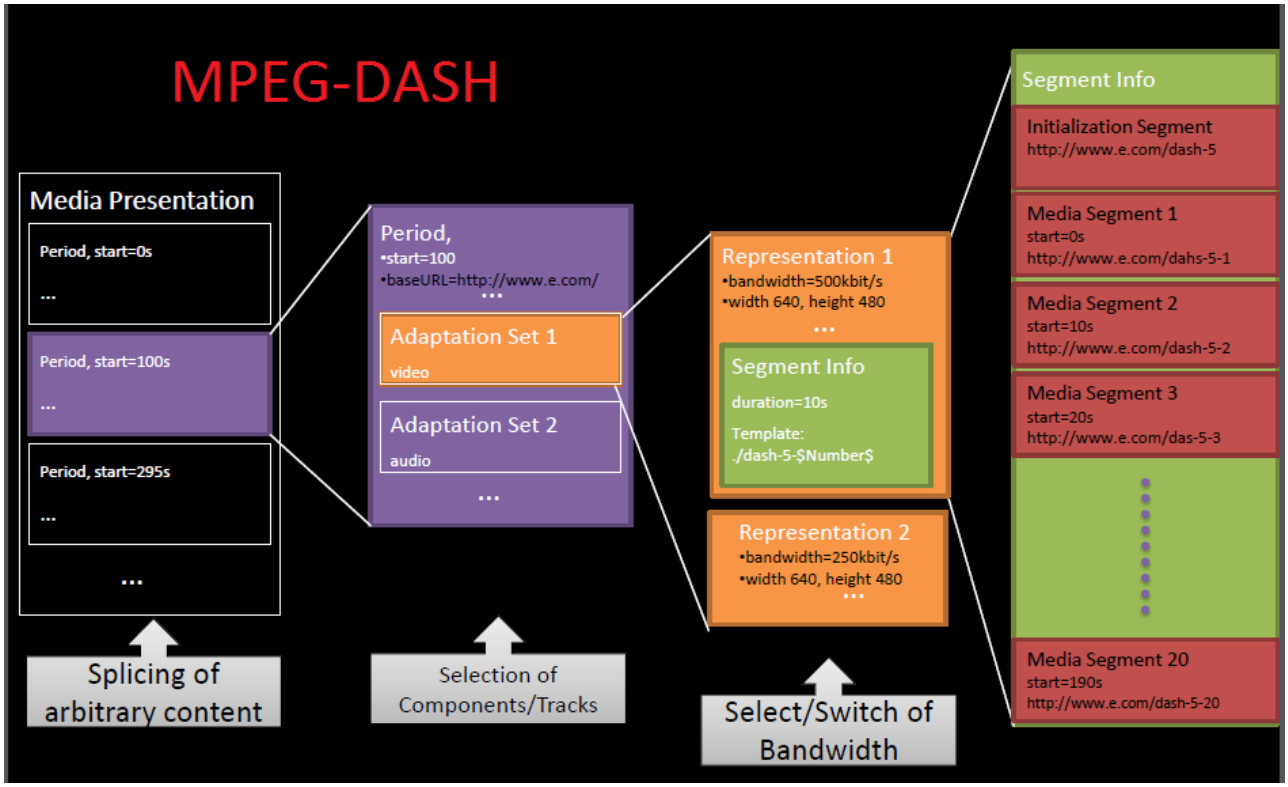


Figura 23. Definición gráfica de segmentos, periodos y representaciones

Como podemos ver una presentación/contenido está compuesto por uno o más periodos, en cada uno de ellos se muestra el tiempo de comienzo. Cada periodo está formado por uno o más "Adaptation Set". Cada set de adaptación se compone de representaciones/calidades del mismo contenido que difieren en el bit-rate y la resolución. Dentro de cada representación están los segmentos.

2.7.4. Direccionamiento URL's

Las url's de cada nivel se resuelven respecto a las url's base del nivel anterior. La información relativa a la URL va ligada tanto a los periodos, como a los sets adaptativos y las representaciones. A veces se proporcionan url's alternativas lo que proporciona redundancia e indica que tenemos acceso al contenido en varias localizaciones.

2.7.5. Funcionamiento

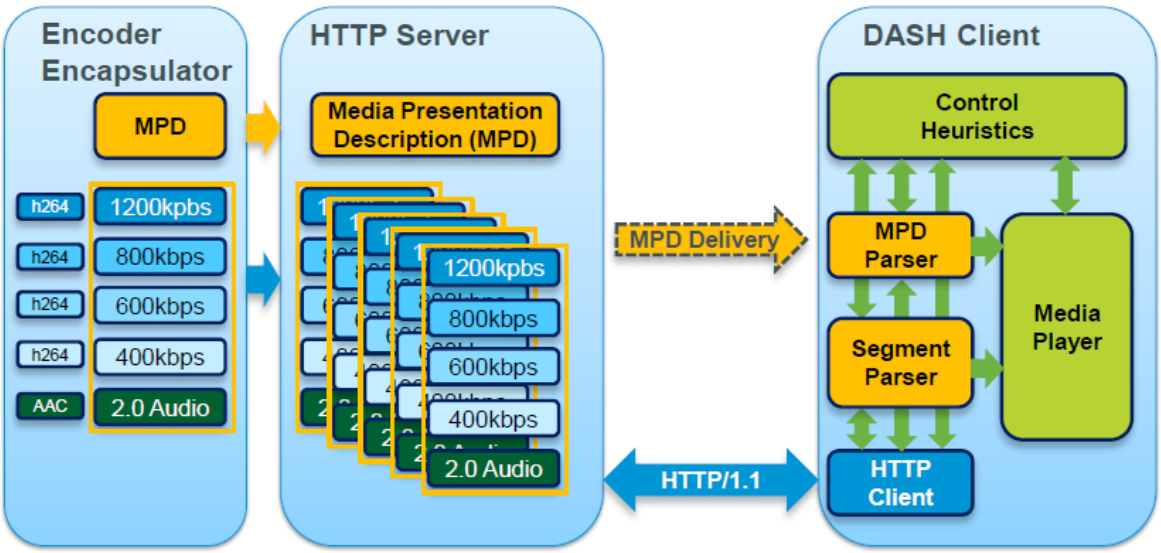


Figura 24. Relación cliente-servidor MPEG-DASH

Como vemos en la figura 24, el codificador genera diferentes calidades del contenido y se deposita el fichero MPD en el servidor. El cliente descarga segmento a segmento el contenido para posteriormente reproducirlo.

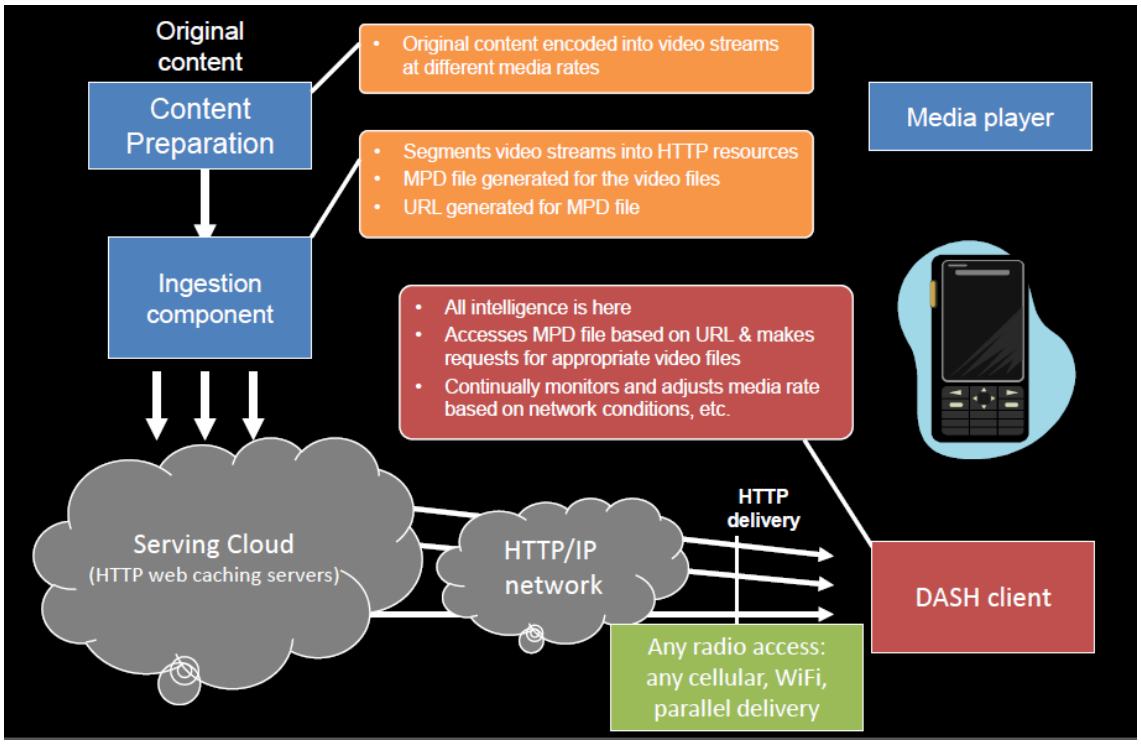


Figura 25. Funcionamiento MPEG-DASH

En la figura 25 podemos ver otro esquema del funcionamiento de MPEG-DASH. El contenido original se codifica a diferentes tasas en el codificador, obteniendo los

distintos streams de vídeo. Posteriormente se generan segmentos (archivos .m4v y .m4a) y el archivo .mpd, que lleva asociado una url pública que nos permita acceder a él. Estos segmentos se depositan en los servidores HTTP pertenecientes a la CDN.

La idea de MPEG-DASH es que el contenido de vídeo se pueda reproducir en cualquier dispositivo.

Como ya hemos comentado el control de la sesión se realiza desde el lado del cliente, que tiene acceso a toda la información de los streams a través de la url pública del .mpd. Es desde el lado del cliente desde dónde se solicitan los cambios de calidades en función de una monitorización que se hace del estado de la red, el uso de recursos del dispositivo, etc.

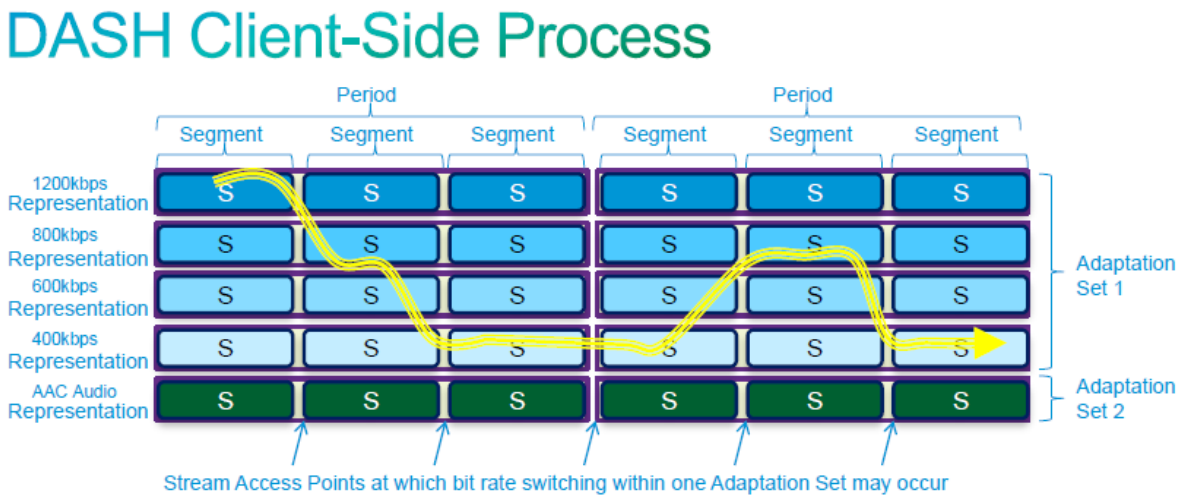


Figura 26. Proceso DASH desde el lado del cliente

Lo primero que hace el cliente es descargar el fichero MPD, posteriormente, descarga los segmentos mediante peticiones http. El bit-rate lo determina siempre el cliente.

2.7.6. Factores para cambio de bit-rate

En MPEG-DASH el cliente es quien determina cuando necesita cambiar la calidad del contenido que está reproduciendo. Algunas de las causas por las que debe realizar el cambio son:

- Condiciones del buffer.
- Estado de la red.
- Cambio de resolución por parte del usuario.
- Carga del dispositivo y uso de recursos.

3. Parte Experimental

3.1. Escenario Global

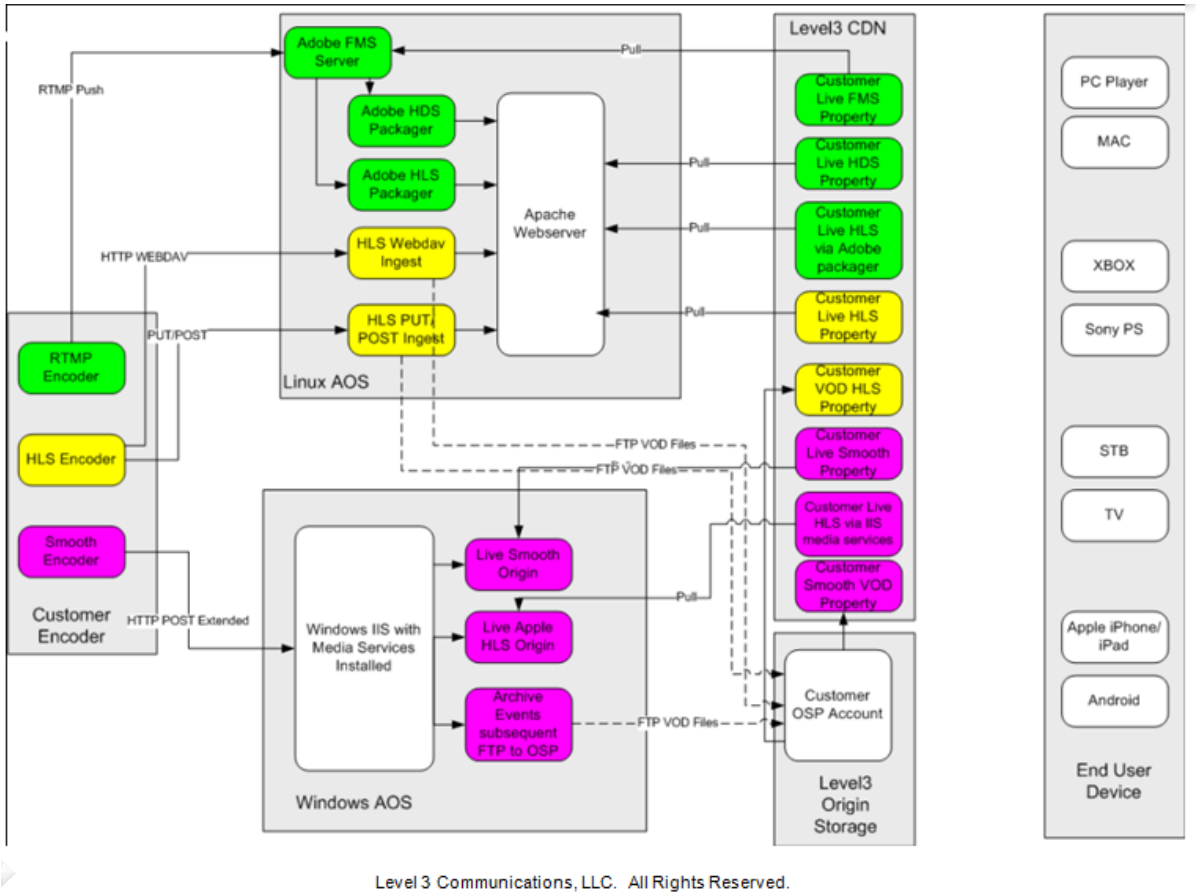


Figura 27. Escenario protocolos de transmisión de vídeo adaptativo

En la figura anterior vemos la estructura de la red sobre la que se va a basar nuestro proyecto. En el bloque de la izquierda se encuentran los codificadores del cliente (nuestros codificadores), en ellos publicamos las propiedades que se solicitan a una empresa proveedora de servicios de CDN y se pueden codificar en tres tipos:

- RTMP: el contenido codificado en RTMP se ingesta en un servidor Linux denominado AOS (Adaptive Origin Server) mediante una acción PUSH. Ahí se trascodifica y multiplexa; y se empaqueta en formato HDS o en formato HLS en un servidor Apache. También se puede obtener el contenido en formato Flash directamente. La CDN, tras recibir una petición de contenido por parte de un usuario, viene a buscar el contenido al AOS y lo entrega al destinatario final para que lo pueda reproducir en un dispositivo. En la CDN tenemos contenido *Live* en los tres formatos: flash, HLS y HDS.

- HLS: el contenido codificado en HLS también se ingesta en un servidor Linux (AOS). En el AOS se configuran los puntos de publicación y la CDN se encarga de buscar el contenido para ofrecer al usuario una propiedad *Live* en formato

HLS. Además, se pueden realizar cortes sobre la propiedad para generar vídeo on demand. Ese contenido se guarda en una cuenta especial que tiene el proveedor de servicios de CDN para almacenar VOD.

- Smooth: Por último también se puede codificar en formato Smooth Streaming. En este caso el contenido se ingesta en un servidor de Windows (AOS). A partir de esta codificación, podemos empaquetar el contenido en Smooth Streaming o en HLS, lo que le permite a la CDN ofrecer contenido *live* con los dos protocolos. Además, también se pueden realizar cortes para ofrecer contenido on demand.

El caso de MPEG-DASH es similar pero se diferencia en algunos aspectos respecto a los tres protocolos anteriores. En nuestro codificador se configura el punto de publicación de la propiedad de MPEG-DASH apuntando por FTP a un servidor Origin perteneciente en este caso a nuestra empresa. Esta propiedad funciona en modo PULL, ya que en este caso es la CDN quien tiene que acudir a nuestro servidor a por el contenido tras recibir una petición del usuario que quiere reproducir dicho contenido.

3.2. Dispositivos a utilizar

En el proyecto se utilizan protocolos orientados a distintos dispositivos, por lo que dependiendo del protocolo de transmisión de vídeo a utilizar, se dispondrá de diferentes escenarios que iremos definiendo según las pruebas a realizar.

Además de todos los elementos que aparecen en la red sobre la que vamos a trabajar y de la que ya hemos hablado anteriormente, utilizaremos una serie de dispositivos dónde podamos reproducir el contenido de vídeo (iPad, TV conectada, Pc portátil).

Con el fin de analizar la robustez de los distintos protocolos utilizaremos un generador de imperfecciones, el Net.Storm [33].

Figura 28. Dispositivo Net.Storm de Albedo

3.3. Escenarios de pruebas

Vamos a dividir las pruebas por protocolos, en nuestro caso, los cuatro estudiados: HLS, Smooth Streaming, HDS y MPEG-DASH. En algunos casos haremos pruebas con propiedades encriptadas que ofrecen algunos operadores de telecomunicaciones y, en otros, emplearemos propiedades que solicitaremos a un proveedor de servicios de CDN (viendo así como es el proceso para poder reproducir vídeo de forma adaptativa por Internet).

Además, procederemos a cambiar ciertos parámetros de los protocolos como la duración de los segmentos o fragmentos, o el DVR, analizando los efectos que provocan sobre los reproductores utilizados con los distintos protocolos.

3.3.1. Pruebas HLS

Para realizar las pruebas con el protocolo HLS hemos utilizado como dispositivo receptor de la señal de vídeo un iPad (Apple). El iPad se conecta a Internet vía wifi y se utiliza como proxy un ordenador portátil. Esto es necesario para poder introducir las imperfecciones con el generador de Albedo Net.Storm. Además con el ordenador es con el que capturamos el tráfico. Se han utilizado dos analizadores de tráfico: el Wireshark [31] y el Fiddler 2 [32], que nos permitirán realizar un análisis del comportamiento de los protocolos.

Por tanto, tenemos el dispositivo Net.Storm conectado a la red de internet y en su otro extremo conectado al ordenador portátil. Éste ejerce de proxy para el iPad, que está conectado al Wifi del laboratorio en el que realizamos las pruebas.

Figura 29. Escenario de pruebas HLS

En primer lugar se han realizado pruebas con las aplicaciones de servicios que están en producción para ver el funcionamiento de los protocolos, para ello se han utilizado los canales de plataformas como ONO o Nubeox (A3), estos servicios están encriptados.

Cada plataforma utiliza un player (reproductor) del que no se ofrecen características de su funcionamiento, y que a través de las pruebas podemos analizar.

Antena 3 utiliza la plataforma Nubeox. Los chunks de HLS tienen una duración de 8 segundos. En cada sub-manifest se reportan 6 chunks y el DVR tiene una duración de 48 segundos. El DVR es la ventana de reproducción y en este caso tiene una configuración fija. La duración de la ventana es proporcional a la duración de un chunk. Dependiendo del player, cuando se descarga la primera playlist, se empieza a reproducir a partir de un cierto punto.

En función de dónde elija el player comenzar la reproducción, tendremos mayor o menor resistencia a cortes. Al final, todo depende de cómo actúe el player, y el corte influirá más o menos según donde se produzca ateniendo a los chunks que tenemos ya descargados de la playlist.

3.3.1.1. Pruebas HLS Nubeox

Como ya hemos comentado los archivos .ts (segmentos) en esta plataforma tienen una duración de 8 segundos. En una reproducción normal (sin cortes) los segmentos se solicitan cada 8 segundos; previamente al envío de cada .ts por parte de la CDN, se realiza una descarga del sub-manifest relativo a la calidad de vídeo que estamos recibiendo.

En primer lugar, provocamos un corte bastante largo para ver cuántos segmentos se almacenan en el buffer. Vemos que el vídeo se reproduce durante unos 15/18 segundos antes de cortarse (dependiendo del momento en el que hagamos el corte), esto equivaldría a unos 2 segmentos aproximadamente.

Si procedemos a insertar una pérdida de duración 8 segundos de tráfico (la duración de un .ts), vemos que la señal de vídeo no se corta, cuando se recupera el tráfico se procede a pedir de forma casi inmediata dos chunks (como se puede ver en la figura 30).

15:55:09.569	15:55:11.897	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams/calle13hd/events/_definst_/liveevent/livestream4Num458902.ts
15:55:17.440	15:55:17.554	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/calle13hd/_definst_/liveevent/livestream4.m3u8?faxs=1
15:55:17.565	15:55:19.184	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams/calle13hd/events/_definst_/liveevent/livestream4Num458903.ts
15:55:25.446	15:55:25.562	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/calle13hd/_definst_/liveevent/livestream4.m3u8?faxs=1
15:55:25.573	15:55:28.095	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams/calle13hd/events/_definst_/liveevent/livestream4Num458904.ts
15:55:37.192	15:55:37.442	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/calle13hd/_definst_/liveevent/livestream4.m3u8?faxs=1
15:55:37.458	15:55:39.813	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams/calle13hd/events/_definst_/liveevent/livestream4Num458905.ts
15:55:41.464	15:55:41.575	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/calle13hd/_definst_/liveevent/livestream4.m3u8?faxs=1
15:55:41.594	15:55:44.228	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams/calle13hd/events/_definst_/liveevent/livestream4Num458906.ts
15:55:49.448	15:55:49.559	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/calle13hd/_definst_/liveevent/livestream4.m3u8?faxs=1
15:55:49.572	15:55:52.797	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams/calle13hd/events/_definst_/liveevent/livestream4Num458907.ts
15:55:57.531	15:55:57.639	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/calle13hd/_definst_/liveevent/livestream4.m3u8?faxs=1
15:55:57.652	15:55:59.971	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams/calle13hd/events/_definst_/liveevent/livestream4Num458908.ts
15:56:05.490	15:56:05.604	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/calle13hd/_definst_/liveevent/livestream4.m3u8?faxs=1
15:56:05.615	15:56:09.298	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams/calle13hd/events/_definst_/liveevent/livestream4Num458909.ts
15:56:13.487	15:56:13.641	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/calle13hd/_definst_/liveevent/livestream4.m3u8?faxs=1
15:56:13.652	15:56:16.822	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams/calle13hd/events/_definst_/liveevent/livestream4Num458910.ts

Figura 30. Captura 1 HLS con la plataforma de Nubeox

El corte se produjo entre las 15:55:28 y las 15:55:36. En apenas 5 segundos se piden dos chunks (los señalados en rojo) y se continúa con la reproducción habitual. En condiciones normales (sin cortes), se va pidiendo un segmento cada 8 segundos.

Al meter una pérdida de duración 16 segundos (correspondiente a 2 chunks), vimos que tampoco se cortaba la reproducción "live". El comportamiento fue que se pidieron 3 archivos .ts de manera consecutiva y se produjo un cambio a una calidad menor de vídeo como se puede apreciar en la siguiente figura.

16:01:41.814	16:01:41.911	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/calle13hd/_definst_/liveevent/livestream4.m3u8?faxs=1
16:01:41.923	16:01:44.215	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams/calle13hd/events/_definst_/liveevent/livestream4Num458951.ts
16:01:49.846	16:01:49.996	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/calle13hd/_definst_/liveevent/livestream4.m3u8?faxs=1
16:01:50.017	16:01:53.473	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams/calle13hd/events/_definst_/liveevent/livestream4Num458952.ts
16:01:57.857	16:01:57.964	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/calle13hd/_definst_/liveevent/livestream4.m3u8?faxs=1
16:01:57.978	16:02:00.460	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams/calle13hd/events/_definst_/liveevent/livestream4Num458953.ts
16:02:05.838	16:02:05.939	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/calle13hd/_definst_/liveevent/livestream4.m3u8?faxs=1
16:02:05.948	16:02:08.646	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams/calle13hd/events/_definst_/liveevent/livestream4Num458954.ts
16:02:27.796	16:02:28.068	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/calle13hd/_definst_/liveevent/livestream3.m3u8?faxs=1
16:02:28.082	16:02:30.100	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams/calle13hd/events/_definst_/liveevent/livestream3Num458954.ts
16:02:31.023	16:02:32.585	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams/calle13hd/events/_definst_/liveevent/livestream3Num458955.ts
16:02:31.193	16:02:31.368	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/calle13hd/_definst_/liveevent/livestream4.m3u8?faxs=1
16:02:32.699	16:02:36.265	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams/calle13hd/events/_definst_/liveevent/livestream3Num458956.ts
16:02:33.950	16:02:34.065	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/calle13hd/_definst_/liveevent/livestream3.m3u8?faxs=1
16:02:37.637	16:02:37.789	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/calle13hd/_definst_/liveevent/livestream4.m3u8?faxs=1
16:02:37.816	16:02:41.050	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams/calle13hd/events/_definst_/liveevent/livestream4Num458955.ts
16:02:41.068	16:02:43.480	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams/calle13hd/events/_definst_/liveevent/livestream4Num458956.ts
16:02:43.501	16:02:45.733	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams/calle13hd/events/_definst_/liveevent/livestream4Num458957.ts
16:02:45.640	16:02:45.775	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/calle13hd/_definst_/liveevent/livestream4.m3u8?faxs=1
16:02:45.796	16:02:47.781	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams/calle13hd/events/_definst_/liveevent/livestream4Num458958.ts
16:02:47.925	16:02:52.350	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams/calle13hd/events/_definst_/liveevent/livestream4Num458959.ts
16:02:53.649	16:02:53.804	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/calle13hd/_definst_/liveevent/livestream4.m3u8?faxs=1
16:02:53.815	16:02:59.382	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams/calle13hd/events/_definst_/liveevent/livestream4Num458960.ts
16:03:01.690	16:03:01.803	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/calle13hd/_definst_/liveevent/livestream4.m3u8?faxs=1
16:03:01.817	16:03:04.555	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams/calle13hd/events/_definst_/liveevent/livestream4Num458961.ts

Figura 31. Captura 2 HLS con la plataforma de Nubeox

En este caso se introdujo un corte desde las 16:02:10 hasta las 16:02:26. Vemos que cuando se recupera el tráfico de vídeo, se mandan tres archivos .ts consecutivos de la calidad 3 ("livestream3"); cuando se estabiliza la situación se vuelve a recuperar la mejor calidad (en este caso, la 4) y se piden los archivos de vídeo necesarios.

Posteriormente, se aumentó el tiempo de pérdidas, y se hicieron diversas pruebas con cortes de 20 y 24 segundos de duración; en todos los casos hubo corte en la reproducción de vídeo y en la mayoría no se recuperó la secuencia de vídeo. Esto nos indica, que el player de Nubeox almacena en buffer dos archivos .ts futuros al "live" de reproducción y que llegados al punto de un corte de duración 24 segundos aproximadamente (duración de tres chunks) no va a poder seguir reproduciendo el contenido sin cortes.

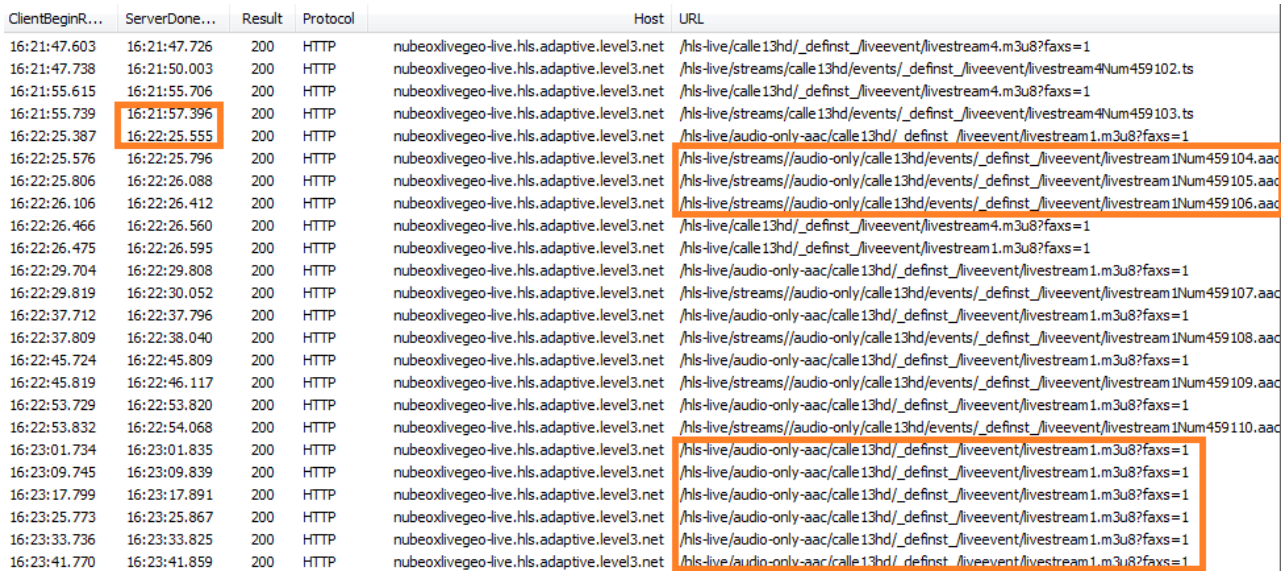
A continuación se muestra un caso en el que hubo corte pero se recuperó el contenido de vídeo sin tener que refrescar el navegador.

ClientBeginR...	ServerDone...	Result	Protocol	Host	URL
16:14:05.394	16:14:05.484	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/calle13hd/_definst_/liveevent/livestream4.m3u8?faxs=1
16:14:05.495	16:14:07.490	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams/calle13hd/events/_definst_/liveevent/livestream4Num459044.ts
16:14:13.377	16:14:13.474	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/calle13hd/_definst_/liveevent/livestream4.m3u8?faxs=1
16:14:13.487	16:14:15.112	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams/calle13hd/events/_definst_/liveevent/livestream4Num459045.ts
16:14:21.410	16:14:21.503	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/calle13hd/_definst_/liveevent/livestream4.m3u8?faxs=1
16:14:21.515	16:14:24.024	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams/calle13hd/events/_definst_/liveevent/livestream4Num459046.ts
16:14:29.419	16:14:29.514	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/calle13hd/_definst_/liveevent/livestream4.m3u8?faxs=1
16:14:29.525	16:14:31.512	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams/calle13hd/events/_definst_/liveevent/livestream4Num459047.ts
16:14:37.436	16:14:37.529	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/calle13hd/_definst_/liveevent/livestream4.m3u8?faxs=1
16:14:37.541	16:14:40.204	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams/calle13hd/events/_definst_/liveevent/livestream4Num459048.ts
16:14:45.438	16:14:45.534	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/calle13hd/_definst_/liveevent/livestream4.m3u8?faxs=1
16:14:45.548	16:14:47.059	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams/calle13hd/events/_definst_/liveevent/livestream4Num459049.ts
16:15:15.361	16:15:15.655	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/calle13hd/_definst_/liveevent/livestream1.m3u8?faxs=1
16:15:15.668	16:15:16.993	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams/calle13hd/events/_definst_/liveevent/livestream1Num459048.ts
16:15:17.014	16:15:17.931	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams/calle13hd/events/_definst_/liveevent/livestream1Num459049.ts
16:15:17.442	16:15:17.637	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/calle13hd/_definst_/liveevent/livestream1.m3u8?faxs=1
16:15:17.982	16:15:19.026	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams/calle13hd/events/_definst_/liveevent/livestream1Num459050.ts
16:15:19.041	16:15:19.974	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams/calle13hd/events/_definst_/liveevent/livestream1Num459051.ts
16:15:19.995	16:15:20.152	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/calle13hd/_definst_/liveevent/livestream4.m3u8?faxs=1
16:15:20.190	16:15:23.002	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams/calle13hd/events/_definst_/liveevent/livestream4Num459050.ts
16:15:20.274	16:15:20.427	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/calle13hd/_definst_/liveevent/livestream4.m3u8?faxs=1
16:15:23.019	16:15:24.839	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams/calle13hd/events/_definst_/liveevent/livestream4Num459051.ts
16:15:24.862	16:15:26.909	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams/calle13hd/events/_definst_/liveevent/livestream4Num459052.ts
16:15:26.926	16:15:28.829	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams/calle13hd/events/_definst_/liveevent/livestream4Num459053.ts
16:15:28.003	16:15:28.110	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/calle13hd/_definst_/liveevent/livestream4.m3u8?faxs=1
16:15:28.844	16:15:30.905	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams/calle13hd/events/_definst_/liveevent/livestream4Num459054.ts
16:15:36.037	16:15:36.148	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/calle13hd/_definst_/liveevent/livestream4.m3u8?faxs=1
16:15:36.159	16:15:38.258	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams/calle13hd/events/_definst_/liveevent/livestream4Num459055.ts
16:15:44.040	16:15:44.189	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/calle13hd/_definst_/liveevent/livestream4.m3u8?faxs=1
16:15:44.200	16:15:45.531	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams/calle13hd/events/_definst_/liveevent/livestream4Num459056.ts

Figura 32. Captura 3 HLS con la plataforma de Nubeox

En este caso introducimos una pérdida desde las 16:14:50 hasta las 16:15:14, la reproducción se detuvo durante un par de segundos, pero tras el corte se vuelve a pedir el manifest y los archivos .ts de la calidad más baja de vídeo ("livestream 1") hasta recuperar la situación normal. Podemos decir que este es el caso deseable ante una imperfección de este tipo, pero experimentalmente vimos que el caso más habitual era el contrario mencionado a continuación.

En la mayoría de ocasiones que realizamos esta prueba (un corte de aproximadamente 24 segundos), la imagen del player se congelaba, pasado el corte se pedían los archivos .aac correspondientes a la calidad más baja (sólo audio) pero el player ya no seguía reproduciendo, ni siquiera audio. Finalmente se solicitaba continuamente el Manifest correspondiente a la calidad más baja pero la reproducción ya no continuaba. A continuación se muestra el comportamiento descrito:



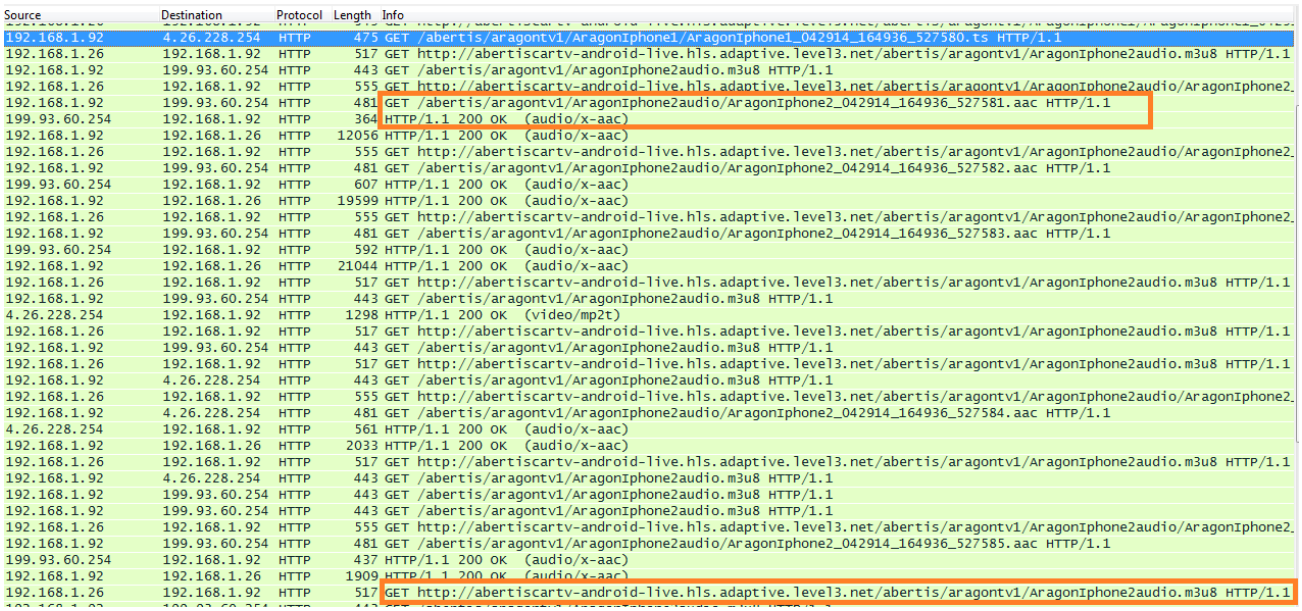
ClientBeginR...	ServerDone...	Result	Protocol	Host	URL
16:21:47.603	16:21:47.726	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/calle13hd/_definst_/liveevent/livestream4.m3u8?faxs=1
16:21:47.738	16:21:50.003	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams/calle13hd/events/_definst_/liveevent/livestream4Num459102.ts
16:21:55.615	16:21:55.706	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/calle13hd/_definst_/liveevent/livestream4.m3u8?faxs=1
16:21:55.739	16:21:57.396	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams/calle13hd/events/_definst_/liveevent/livestream4Num459103.ts
16:22:25.387	16:22:25.555	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/audio-only-aac/calle13hd/_definst_/liveevent/livestream1.m3u8?faxs=1
16:22:25.576	16:22:25.796	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams//audio-only/calle13hd/events/_definst_/liveevent/livestream1Num459104.aac
16:22:25.806	16:22:26.088	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams//audio-only/calle13hd/events/_definst_/liveevent/livestream1Num459105.aac
16:22:26.106	16:22:26.412	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams//audio-only/calle13hd/events/_definst_/liveevent/livestream1Num459106.aac
16:22:26.466	16:22:26.560	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/calle13hd/_definst_/liveevent/livestream4.m3u8?faxs=1
16:22:26.475	16:22:26.595	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/calle13hd/_definst_/liveevent/livestream1.m3u8?faxs=1
16:22:29.704	16:22:29.808	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/audio-only-aac/calle13hd/_definst_/liveevent/livestream1.m3u8?faxs=1
16:22:29.819	16:22:30.052	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams//audio-only/calle13hd/events/_definst_/liveevent/livestream1Num459107.aac
16:22:37.712	16:22:37.796	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/audio-only-aac/calle13hd/_definst_/liveevent/livestream1.m3u8?faxs=1
16:22:37.809	16:22:38.040	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams//audio-only/calle13hd/events/_definst_/liveevent/livestream1Num459108.aac
16:22:45.724	16:22:45.809	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/audio-only-aac/calle13hd/_definst_/liveevent/livestream1.m3u8?faxs=1
16:22:45.819	16:22:46.117	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams//audio-only/calle13hd/events/_definst_/liveevent/livestream1Num459109.aac
16:22:53.729	16:22:53.820	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/audio-only-aac/calle13hd/_definst_/liveevent/livestream1.m3u8?faxs=1
16:22:53.832	16:22:54.068	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/streams//audio-only/calle13hd/events/_definst_/liveevent/livestream1Num459110.aac
16:23:01.734	16:23:01.835	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/audio-only-aac/calle13hd/_definst_/liveevent/livestream1.m3u8?faxs=1
16:23:09.745	16:23:09.839	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/audio-only-aac/calle13hd/_definst_/liveevent/livestream1.m3u8?faxs=1
16:23:17.799	16:23:17.891	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/audio-only-aac/calle13hd/_definst_/liveevent/livestream1.m3u8?faxs=1
16:23:25.773	16:23:25.867	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/audio-only-aac/calle13hd/_definst_/liveevent/livestream1.m3u8?faxs=1
16:23:33.736	16:23:33.825	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/audio-only-aac/calle13hd/_definst_/liveevent/livestream1.m3u8?faxs=1
16:23:41.770	16:23:41.859	200	HTTP	nubeoxlivegeo-live.hls.adaptive.level3.net	/hls-live/audio-only-aac/calle13hd/_definst_/liveevent/livestream1.m3u8?faxs=1

Figura 33. Captura 4 HLS con la plataforma de Nubeox

La única manera de recuperar el vídeo es refrescar manualmente el navegador, o en este caso el reproductor de Nubeox (este es un comportamiento no deseable pero que ocurre dentro de este tipo de players).

Por tanto, cuando los cortes introducidos no afectan a la reproducción, el efecto es que se piden archivos .ts de forma más consecutiva y puede haber cambios de calidades. Mientras que cuando los cortes sobrepasan el tamaño del buffer, tenemos un congelado de la imagen y en caso de recuperar la reproducción se comienza por la calidad más baja.

Cabe destacar que se realizaron las mismas pruebas con un canal en abierto (no encriptado) y se obtuvieron resultados muy similares bajo la misma situación. A continuación mostramos una captura realizada con el software Wireshark, dónde se puede observar que tras el corte se solicitan los archivos de audio, la CDN (*Content Delivery Network*) devuelve dichos archivos (respuesta 200 OK) pero el reproductor no es capaz de reanudar el contenido de vídeo. En esta captura podemos ver las peticiones del iPad (con ip 192.168.1.26) al proxy (con ip 192.168.1.92) y del proxy a la CDN (con ip 4.26.228.254).



Source	Destination	Protocol	Length	Info
192.168.1.92	4.26.228.254	HTTP	475	GET /abertis/aragontv1/AragonIphone1/AragonIphone1_042914_164936_527580.ts HTTP/1.1
192.168.1.26	192.168.1.92	HTTP	517	GET http://abertiscartv-android-live.hls.adaptive.level3.net/abertis/aragontv1/AragonIphone2audio.m3u8 HTTP/1.1
192.168.1.92	199.93.60.254	HTTP	443	GET /abertis/aragontv1/AragonIphone2audio.m3u8 HTTP/1.1
192.168.1.26	192.168.1.92	HTTP	555	GET http://abertiscartv-android-live.hls.adaptive.level3.net/abertis/aragontv1/AragonIphone2audio/AragonIphone2_
192.168.1.92	199.93.60.254	HTTP	481	GET /abertis/aragontv1/AragonIphone2audio/AragonIphone2_042914_164936_527581.aac HTTP/1.1
199.93.60.254	192.168.1.92	HTTP	364	HTTP/1.1 200 OK (audio/x-aac)
192.168.1.92	192.168.1.26	HTTP	12056	HTTP/1.1 200 OK (audio/x-aac)
192.168.1.26	192.168.1.92	HTTP	555	GET http://abertiscartv-android-live.hls.adaptive.level3.net/abertis/aragontv1/AragonIphone2audio/AragonIphone2_
192.168.1.92	199.93.60.254	HTTP	481	GET /abertis/aragontv1/AragonIphone2audio/AragonIphone2_042914_164936_527582.aac HTTP/1.1
199.93.60.254	192.168.1.92	HTTP	607	HTTP/1.1 200 OK (audio/x-aac)
192.168.1.92	192.168.1.26	HTTP	19599	HTTP/1.1 200 OK (audio/x-aac)
192.168.1.26	192.168.1.92	HTTP	555	GET http://abertiscartv-android-live.hls.adaptive.level3.net/abertis/aragontv1/AragonIphone2audio/AragonIphone2_
192.168.1.92	199.93.60.254	HTTP	481	GET /abertis/aragontv1/AragonIphone2audio/AragonIphone2_042914_164936_527583.aac HTTP/1.1
199.93.60.254	192.168.1.92	HTTP	592	HTTP/1.1 200 OK (audio/x-aac)
192.168.1.92	192.168.1.26	HTTP	21044	HTTP/1.1 200 OK (audio/x-aac)
192.168.1.26	192.168.1.92	HTTP	517	GET http://abertiscartv-android-live.hls.adaptive.level3.net/abertis/aragontv1/AragonIphone2audio.m3u8 HTTP/1.1
192.168.1.92	199.93.60.254	HTTP	443	GET /abertis/aragontv1/AragonIphone2audio.m3u8 HTTP/1.1
4.26.228.254	192.168.1.92	HTTP	1298	HTTP/1.1 200 OK (video/mp2t)
192.168.1.26	192.168.1.92	HTTP	517	GET http://abertiscartv-android-live.hls.adaptive.level3.net/abertis/aragontv1/AragonIphone2audio.m3u8 HTTP/1.1
192.168.1.92	199.93.60.254	HTTP	443	GET /abertis/aragontv1/AragonIphone2audio.m3u8 HTTP/1.1
192.168.1.26	192.168.1.92	HTTP	517	GET http://abertiscartv-android-live.hls.adaptive.level3.net/abertis/aragontv1/AragonIphone2audio.m3u8 HTTP/1.1
192.168.1.92	4.26.228.254	HTTP	443	GET /abertis/aragontv1/AragonIphone2audio.m3u8 HTTP/1.1
192.168.1.26	192.168.1.92	HTTP	555	GET http://abertiscartv-android-live.hls.adaptive.level3.net/abertis/aragontv1/AragonIphone2audio/AragonIphone2_
192.168.1.92	4.26.228.254	HTTP	481	GET /abertis/aragontv1/AragonIphone2audio/AragonIphone2_042914_164936_527584.aac HTTP/1.1
4.26.228.254	192.168.1.92	HTTP	561	HTTP/1.1 200 OK (audio/x-aac)
192.168.1.92	192.168.1.26	HTTP	2033	HTTP/1.1 200 OK (audio/x-aac)
192.168.1.26	192.168.1.92	HTTP	517	GET http://abertiscartv-android-live.hls.adaptive.level3.net/abertis/aragontv1/AragonIphone2audio.m3u8 HTTP/1.1
192.168.1.92	4.26.228.254	HTTP	443	GET /abertis/aragontv1/AragonIphone2audio.m3u8 HTTP/1.1
192.168.1.92	199.93.60.254	HTTP	443	GET /abertis/aragontv1/AragonIphone2audio.m3u8 HTTP/1.1
192.168.1.92	199.93.60.254	HTTP	443	GET /abertis/aragontv1/AragonIphone2audio.m3u8 HTTP/1.1
192.168.1.26	192.168.1.92	HTTP	555	GET http://abertiscartv-android-live.hls.adaptive.level3.net/abertis/aragontv1/AragonIphone2audio/AragonIphone2_
192.168.1.92	199.93.60.254	HTTP	481	GET /abertis/aragontv1/AragonIphone2audio/AragonIphone2_042914_164936_527585.aac HTTP/1.1
199.93.60.254	192.168.1.92	HTTP	437	HTTP/1.1 200 OK (audio/x-aac)
192.168.1.92	192.168.1.26	HTTP	1909	HTTP/1.1 200 OK (audio/x-aac)
192.168.1.26	192.168.1.92	HTTP	517	GET http://abertiscartv-android-live.hls.adaptive.level3.net/abertis/aragontv1/AragonIphone2audio.m3u8 HTTP/1.1

Figura 34. Captura canal HLS en abierto

Finalmente el iPad se queda pidiendo el Manifest pero la CDN ya no responde. Como hemos dicho para reanudar la reproducción el usuario debe reiniciar manualmente la aplicación.

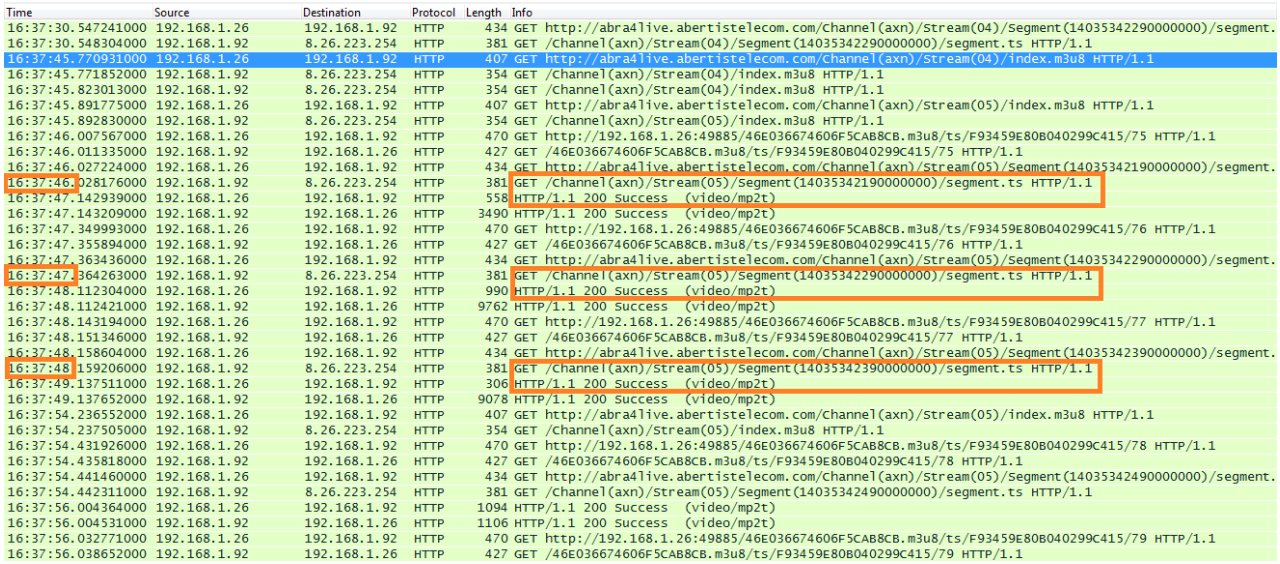
3.3.1.2. Pruebas HLS Ono

Los chunks de HLS en Ono tienen una duración de 10 segundos, que es la típica estandarizada de este protocolo. En cada submanifest aparecen 12 segmentos (archivos .ts) por lo que el DVR es de 120 segundos.

Procedemos a realizar pruebas similares a las llevadas a cabo para el player de Nubeox. Las diferencias con el caso anterior es que la duración de los segmentos es distinta y el DVR también es de diferente duración. En una reproducción normal (sin imperfecciones) los chunks se transmiten cada 10 segundos y tenemos una ventana de reproducción de 120 segundos (para retroceder) para el contenido “live”.

Si provocamos un corte prolongado para ver cuántos segmentos se almacenan en el buffer, observamos que la reproducción se corta pasados unos 24 segundos a partir del corte (se hicieron varias pruebas obteniendo tiempos entre 20 y 30 segundos). Por tanto, en el buffer se guardan 2/3 segmentos de 10 segundos cada uno.

A continuación, introducimos un corte de 10 segundos, desde las 16:37:35 hasta las 16:37:45.



Time	Source	Destination	Protocol	Length	Info
16:37:30.547241000	192.168.1.26	192.168.1.92	HTTP	434	GET http://abra4live.abertistelecom.com/Channel(axn)/Stream(04)/Segment(14035342290000000)/segment.
16:37:30.548304000	192.168.1.92	8.26.223.254	HTTP	381	GET /Channel(axn)/Stream(04)/Segment(14035342290000000)/segment.ts HTTP/1.1
16:37:45.770931000	192.168.1.26	192.168.1.92	HTTP	407	GET http://abra4live.abertistelecom.com/Channel(axn)/Stream(04)/index.m3u8 HTTP/1.1
16:37:45.771852000	192.168.1.92	8.26.223.254	HTTP	354	GET /Channel(axn)/Stream(04)/index.m3u8 HTTP/1.1
16:37:45.823013000	192.168.1.92	8.26.223.254	HTTP	354	GET /Channel(axn)/Stream(04)/index.m3u8 HTTP/1.1
16:37:45.891775000	192.168.1.26	192.168.1.92	HTTP	407	GET http://abra4live.abertistelecom.com/Channel(axn)/Stream(05)/index.m3u8 HTTP/1.1
16:37:45.892830000	192.168.1.92	8.26.223.254	HTTP	354	GET /Channel(axn)/Stream(05)/index.m3u8 HTTP/1.1
16:37:46.007567000	192.168.1.26	192.168.1.92	HTTP	470	GET http://192.168.1.26:49885/46E036674606F5CAB8CB.m3u8/ts/F93459E80B040299C415/75 HTTP/1.1
16:37:46.011335000	192.168.1.92	192.168.1.26	HTTP	427	GET /46E036674606F5CAB8CB.m3u8/ts/F93459E80B040299C415/75 HTTP/1.1
16:37:46.027224000	192.168.1.26	192.168.1.92	HTTP	434	GET http://abra4live.abertistelecom.com/Channel(axn)/Stream(05)/Segment(14035342190000000)/segment.
16:37:46.028176000	192.168.1.92	8.26.223.254	HTTP	381	GET /Channel(axn)/Stream(05)/Segment(14035342190000000)/segment.ts HTTP/1.1
16:37:47.142939000	192.168.1.26	192.168.1.92	HTTP	558	HTTP/1.1 200 Success (video/mp2t)
16:37:47.143209000	192.168.1.92	192.168.1.26	HTTP	3490	HTTP/1.1 200 Success (video/mp2t)
16:37:47.349993000	192.168.1.26	192.168.1.92	HTTP	470	GET http://192.168.1.26:49885/46E036674606F5CAB8CB.m3u8/ts/F93459E80B040299C415/76 HTTP/1.1
16:37:47.355894000	192.168.1.92	192.168.1.26	HTTP	427	GET /46E036674606F5CAB8CB.m3u8/ts/F93459E80B040299C415/76 HTTP/1.1
16:37:47.363436000	192.168.1.26	192.168.1.92	HTTP	434	GET http://abra4live.abertistelecom.com/Channel(axn)/Stream(05)/Segment(14035342290000000)/segment.
16:37:47.364263000	192.168.1.92	8.26.223.254	HTTP	381	GET /Channel(axn)/Stream(05)/Segment(14035342290000000)/segment.ts HTTP/1.1
16:37:48.112304000	192.168.1.92	192.168.1.26	HTTP	990	HTTP/1.1 200 Success (video/mp2t)
16:37:48.112421000	192.168.1.92	192.168.1.26	HTTP	9762	HTTP/1.1 200 Success (video/mp2t)
16:37:48.143194000	192.168.1.26	192.168.1.92	HTTP	470	GET http://192.168.1.26:49885/46E036674606F5CAB8CB.m3u8/ts/F93459E80B040299C415/77 HTTP/1.1
16:37:48.151346000	192.168.1.92	192.168.1.26	HTTP	427	GET /46E036674606F5CAB8CB.m3u8/ts/F93459E80B040299C415/77 HTTP/1.1
16:37:48.158604000	192.168.1.26	192.168.1.92	HTTP	434	GET http://abra4live.abertistelecom.com/Channel(axn)/Stream(05)/Segment(14035342390000000)/segment.
16:37:48.159206000	192.168.1.92	8.26.223.254	HTTP	381	GET /Channel(axn)/Stream(05)/Segment(14035342390000000)/segment.ts HTTP/1.1
16:37:49.137511000	192.168.1.26	192.168.1.92	HTTP	306	HTTP/1.1 200 Success (video/mp2t)
16:37:49.137652000	192.168.1.92	192.168.1.26	HTTP	9078	HTTP/1.1 200 Success (video/mp2t)
16:37:54.236552000	192.168.1.26	192.168.1.92	HTTP	407	GET http://abra4live.abertistelecom.com/Channel(axn)/Stream(05)/index.m3u8 HTTP/1.1
16:37:54.237505000	192.168.1.92	8.26.223.254	HTTP	354	GET /Channel(axn)/Stream(05)/index.m3u8 HTTP/1.1
16:37:54.431926000	192.168.1.26	192.168.1.92	HTTP	470	GET http://192.168.1.26:49885/46E036674606F5CAB8CB.m3u8/ts/F93459E80B040299C415/78 HTTP/1.1
16:37:54.435818000	192.168.1.92	192.168.1.26	HTTP	427	GET /46E036674606F5CAB8CB.m3u8/ts/F93459E80B040299C415/78 HTTP/1.1
16:37:54.441460000	192.168.1.26	192.168.1.92	HTTP	434	GET http://abra4live.abertistelecom.com/Channel(axn)/Stream(05)/Segment(14035342490000000)/segment.
16:37:54.442311000	192.168.1.92	8.26.223.254	HTTP	381	GET /Channel(axn)/Stream(05)/Segment(14035342490000000)/segment.ts HTTP/1.1
16:37:56.004364000	192.168.1.26	192.168.1.92	HTTP	1094	HTTP/1.1 200 Success (video/mp2t)
16:37:56.004531000	192.168.1.92	192.168.1.26	HTTP	1106	HTTP/1.1 200 Success (video/mp2t)
16:37:56.032771000	192.168.1.26	192.168.1.92	HTTP	470	GET http://192.168.1.26:49885/46E036674606F5CAB8CB.m3u8/ts/F93459E80B040299C415/79 HTTP/1.1
16:37:56.038652000	192.168.1.92	192.168.1.26	HTTP	427	GET /46E036674606F5CAB8CB.m3u8/ts/F93459E80B040299C415/79 HTTP/1.1

Figura 35. Captura 1 HLS con la plataforma de Ono

El comportamiento es similar al caso de Nubeox, no hay corte en la reproducción, y cuando acaba el corte programado se piden varios archivos de vídeo de forma consecutiva.

Posteriormente se introduce un corte de 20 segundos, desde las 16:41:50 hasta las 16:42:10. De nuevo no tenemos cortes en el vídeo; se aprecia algún cambio de calidad, puesto que se piden diferentes sub-manifest. Cabe destacar que en HLS se pide el sub-manifest de forma repetitiva, cosa que no ocurre en otros protocolos como HDS y Smooth Streaming como veremos más tarde.

Ante cortes mayores, refleja el mismo comportamiento que para el player de Nubeox, la reproducción se corta y ya no se reanuda.

3.3.1.3. Pruebas HLS Safari

Hemos realizados estas pruebas con una propiedad pública en el navegador de Safari del iPad y el modo de funcionamiento es el mismo que en los casos anteriores, obteniendo resultados similares. Aún así, como es una propiedad no encriptada, vamos a detallar el análisis como anteriormente.

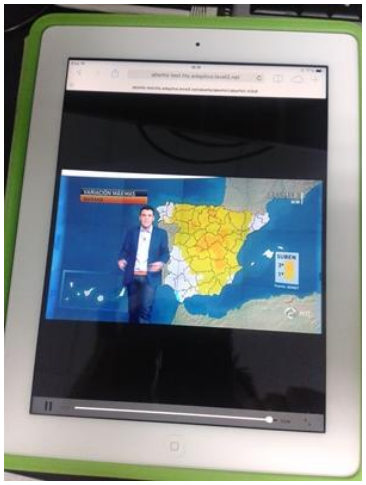


Figura 36. Reproductor de Safari en el iPad para pruebas HLS

La propiedad que utilizamos tiene 3 calidades (una de audio y dos de vídeo), con un DVR de 120 segundos.

Si nos descargamos el sub-manifest de una de las calidades, obtenemos lo siguiente:

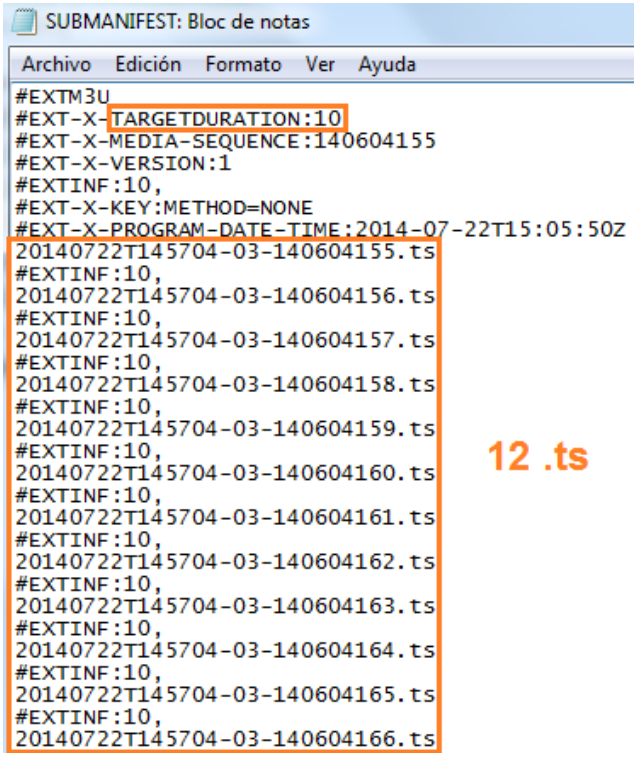


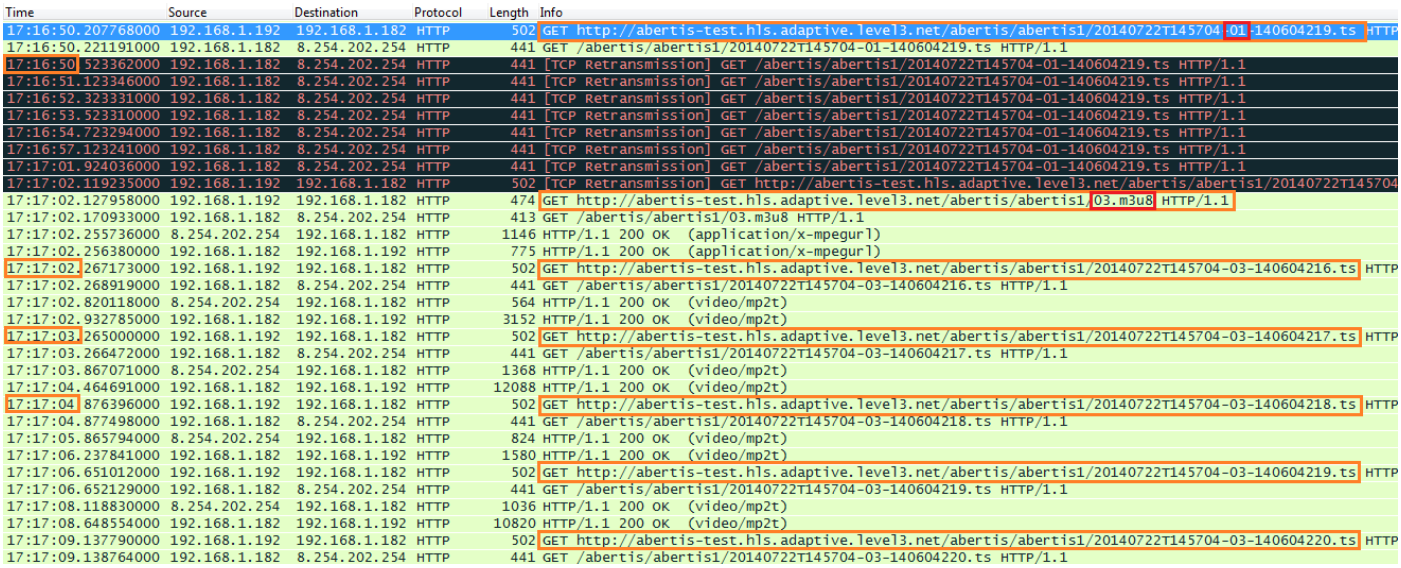
Figura 37. Sub-Manifest Propiedad HLS de Safari

Podemos ver que la duración de los segmentos es de 10 segundos y que tenemos 12 .ts en cada sub-manifest, número de .ts necesarios para cubrir la duración del DVR.

Para comenzar realizamos un corte prolongado y vemos que la reproducción continúa durante aproximadamente 20 segundos desde el inicio del corte. Esto equivale a unos 2

segmentos. Por tanto, en el buffer se almacenan unos 2 segmentos de 10 segundos (similar a los casos anteriores).

Si introducimos un corte en el tráfico de 10 segundos de duración, desde las 17:16:50 hasta las 17:17:00, la reproducción no se corta como ha ocurrido también anteriormente.

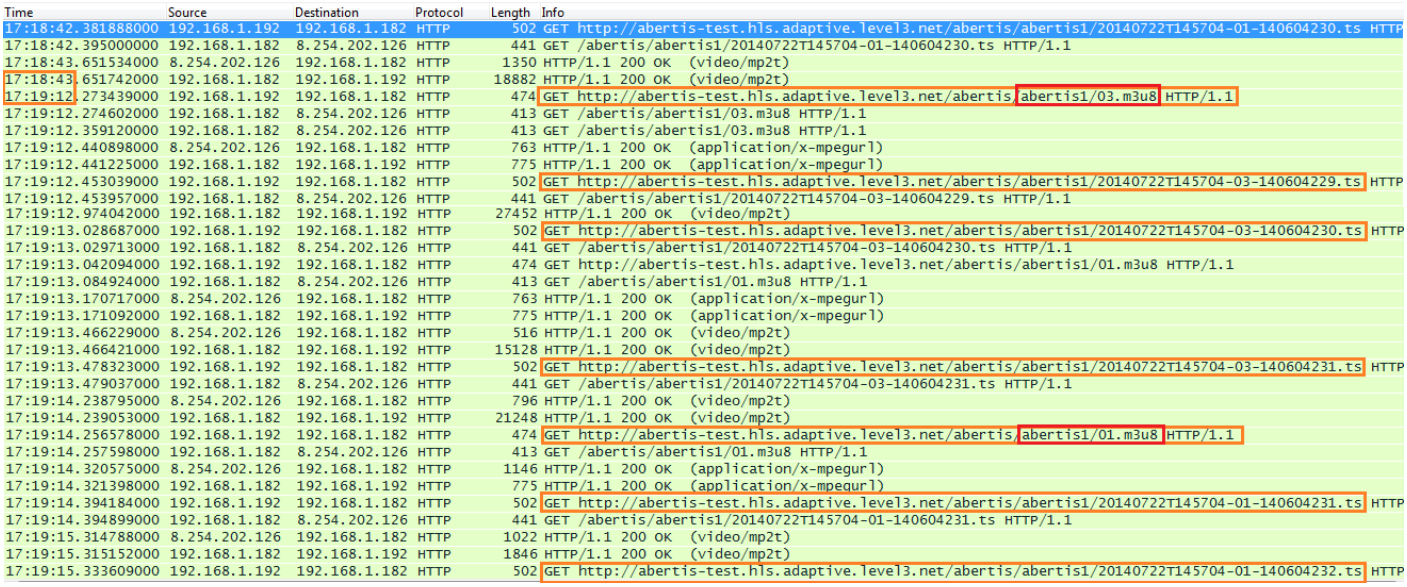


Time	Source	Destination	Protocol	Length	Info
17:16:50.207768000	192.168.1.192	192.168.1.182	HTTP	502	GET http://abertis-test.hls.adaptive.level3.net/abertis/abertis1/20140722T145704-01-140604219.ts HTTP
17:16:50.221191000	192.168.1.182	8.254.202.254	HTTP	441	GET /abertis/abertis1/20140722T145704-01-140604219.ts HTTP/1.1
17:16:50.523362000	192.168.1.182	8.254.202.254	HTTP	441	[TCP Retransmission] GET /abertis/abertis1/20140722T145704-01-140604219.ts HTTP/1.1
17:16:51.123346000	192.168.1.182	8.254.202.254	HTTP	441	[TCP Retransmission] GET /abertis/abertis1/20140722T145704-01-140604219.ts HTTP/1.1
17:16:52.323331000	192.168.1.182	8.254.202.254	HTTP	441	[TCP Retransmission] GET /abertis/abertis1/20140722T145704-01-140604219.ts HTTP/1.1
17:16:53.523310000	192.168.1.182	8.254.202.254	HTTP	441	[TCP Retransmission] GET /abertis/abertis1/20140722T145704-01-140604219.ts HTTP/1.1
17:16:54.723294000	192.168.1.182	8.254.202.254	HTTP	441	[TCP Retransmission] GET /abertis/abertis1/20140722T145704-01-140604219.ts HTTP/1.1
17:16:57.123241000	192.168.1.182	8.254.202.254	HTTP	441	[TCP Retransmission] GET /abertis/abertis1/20140722T145704-01-140604219.ts HTTP/1.1
17:17:01.924036000	192.168.1.182	8.254.202.254	HTTP	441	[TCP Retransmission] GET /abertis/abertis1/20140722T145704-01-140604219.ts HTTP/1.1
17:17:02.119235000	192.168.1.192	192.168.1.182	HTTP	502	[TCP Retransmission] GET http://abertis-test.hls.adaptive.level3.net/abertis/abertis1/20140722T145704
17:17:02.127958000	192.168.1.192	192.168.1.182	HTTP	474	GET http://abertis-test.hls.adaptive.level3.net/abertis/abertis1/03.m3u8 HTTP/1.1
17:17:02.170933000	192.168.1.182	8.254.202.254	HTTP	413	GET /abertis/abertis1/03.m3u8 HTTP/1.1
17:17:02.255736000	8.254.202.254	192.168.1.182	HTTP	1146	HTTP/1.1 200 OK (application/x-mpegurl)
17:17:02.256380000	192.168.1.182	192.168.1.192	HTTP	775	HTTP/1.1 200 OK (application/x-mpegurl)
17:17:02.267173000	192.168.1.192	192.168.1.182	HTTP	502	GET http://abertis-test.hls.adaptive.level3.net/abertis/abertis1/20140722T145704-03-140604216.ts HTTP
17:17:02.268919000	192.168.1.182	8.254.202.254	HTTP	441	GET /abertis/abertis1/20140722T145704-03-140604216.ts HTTP/1.1
17:17:02.820118000	8.254.202.254	192.168.1.182	HTTP	564	HTTP/1.1 200 OK (video/mp2t)
17:17:02.932785000	192.168.1.182	192.168.1.192	HTTP	3152	HTTP/1.1 200 OK (video/mp2t)
17:17:03.265000000	192.168.1.192	192.168.1.182	HTTP	502	GET http://abertis-test.hls.adaptive.level3.net/abertis/abertis1/20140722T145704-03-140604217.ts HTTP
17:17:03.266472000	192.168.1.182	8.254.202.254	HTTP	441	GET /abertis/abertis1/20140722T145704-03-140604217.ts HTTP/1.1
17:17:03.867071000	8.254.202.254	192.168.1.182	HTTP	1368	HTTP/1.1 200 OK (video/mp2t)
17:17:04.464691000	192.168.1.182	192.168.1.192	HTTP	12088	HTTP/1.1 200 OK (video/mp2t)
17:17:04.876396000	192.168.1.192	192.168.1.182	HTTP	502	GET http://abertis-test.hls.adaptive.level3.net/abertis/abertis1/20140722T145704-03-140604218.ts HTTP
17:17:04.877498000	192.168.1.182	8.254.202.254	HTTP	441	GET /abertis/abertis1/20140722T145704-03-140604218.ts HTTP/1.1
17:17:05.865794000	8.254.202.254	192.168.1.182	HTTP	824	HTTP/1.1 200 OK (video/mp2t)
17:17:06.237841000	192.168.1.182	192.168.1.192	HTTP	1580	HTTP/1.1 200 OK (video/mp2t)
17:17:06.651012000	192.168.1.192	192.168.1.182	HTTP	502	GET http://abertis-test.hls.adaptive.level3.net/abertis/abertis1/20140722T145704-03-140604219.ts HTTP
17:17:06.652129000	192.168.1.182	8.254.202.254	HTTP	441	GET /abertis/abertis1/20140722T145704-03-140604219.ts HTTP/1.1
17:17:08.118830000	8.254.202.254	192.168.1.182	HTTP	1036	HTTP/1.1 200 OK (video/mp2t)
17:17:08.648554000	192.168.1.182	192.168.1.192	HTTP	10820	HTTP/1.1 200 OK (video/mp2t)
17:17:09.137790000	192.168.1.192	192.168.1.182	HTTP	502	GET http://abertis-test.hls.adaptive.level3.net/abertis/abertis1/20140722T145704-03-140604220.ts HTTP
17:17:09.138764000	192.168.1.182	8.254.202.254	HTTP	441	GET /abertis/abertis1/20140722T145704-03-140604220.ts HTTP/1.1

Figura 38. Captura 1 HLS propiedad de pruebas

Vemos de nuevo que cuando se recupera la comunicación, se solicita el manifest (03.m3u8), en este caso de una calidad más baja a la que se estaba reproduciendo antes del corte; y se piden varios segmentos de manera consecutiva (en pocos segundos). Posteriormente vuelve a haber un cambio de calidad, se recupera la calidad más alta, y hasta transcurridos 20 segundos de la recuperación no se vuelve a la situación normal de enviar segmentos cada 10 segundos.

Si realizamos un corte de 20 segundos, en este caso desde las 17:18:50 hasta las 17:19:10, tenemos un corte de apenas 1 segundo y se reanuda el vídeo. Al capturar el tráfico obtenemos unos resultados muy similares al anterior.

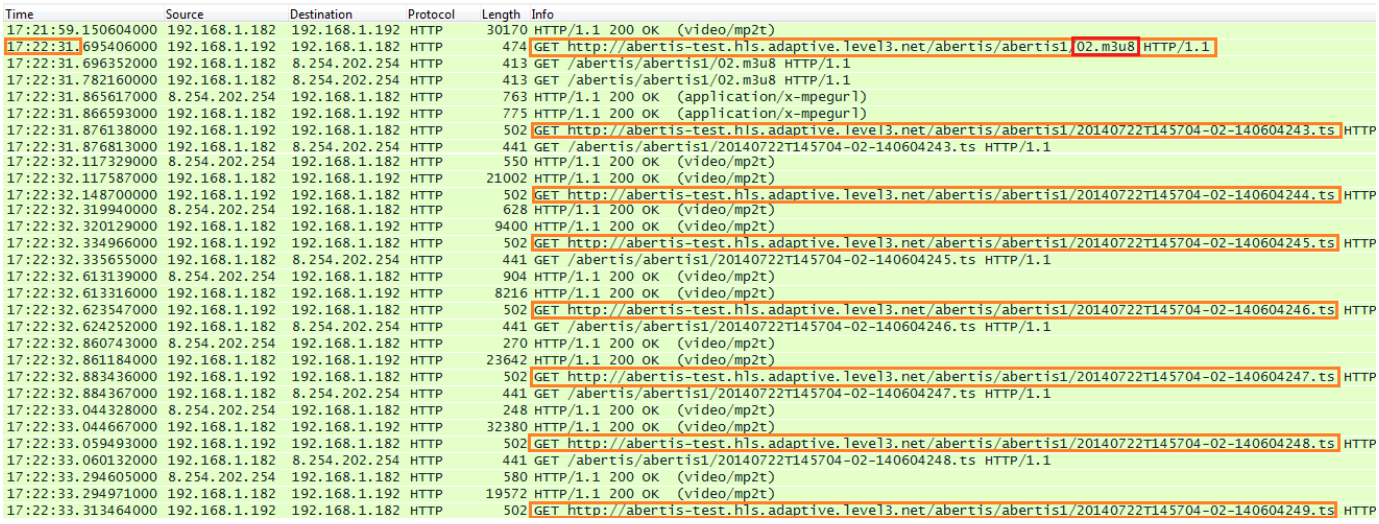


Time	Source	Destination	Protocol	Length	Info
17:18:42.381888000	192.168.1.192	192.168.1.182	HTTP	502	GET http://abertis-test.hls.adaptive.level3.net/abertis/abertis1/20140722T145704-01-140604230.ts HTTP
17:18:42.395000000	192.168.1.182	8.254.202.126	HTTP	441	GET /abertis/abertis1/20140722T145704-01-140604230.ts HTTP/1.1
17:18:43.651534000	8.254.202.126	192.168.1.182	HTTP	1350	HTTP/1.1 200 OK (video/mp2t)
17:18:43.651742000	192.168.1.182	192.168.1.192	HTTP	18882	HTTP/1.1 200 OK (video/mp2t)
17:19:12.273439000	192.168.1.192	192.168.1.182	HTTP	474	GET http://abertis-test.hls.adaptive.level3.net/abertis/abertis1/03.m3u8 HTTP/1.1
17:19:12.274602000	192.168.1.182	8.254.202.126	HTTP	413	GET /abertis/abertis1/03.m3u8 HTTP/1.1
17:19:12.359120000	192.168.1.182	8.254.202.126	HTTP	413	GET /abertis/abertis1/03.m3u8 HTTP/1.1
17:19:12.440898000	8.254.202.126	192.168.1.182	HTTP	763	HTTP/1.1 200 OK (application/x-mpegurl)
17:19:12.441225000	192.168.1.182	192.168.1.192	HTTP	775	HTTP/1.1 200 OK (application/x-mpegurl)
17:19:12.453039000	192.168.1.192	192.168.1.182	HTTP	502	GET http://abertis-test.hls.adaptive.level3.net/abertis/abertis1/20140722T145704-03-140604229.ts HTTP
17:19:12.453957000	192.168.1.182	8.254.202.126	HTTP	441	GET /abertis/abertis1/20140722T145704-03-140604229.ts HTTP/1.1
17:19:12.974042000	192.168.1.182	192.168.1.192	HTTP	27452	HTTP/1.1 200 OK (video/mp2t)
17:19:13.028687000	192.168.1.192	192.168.1.182	HTTP	502	GET http://abertis-test.hls.adaptive.level3.net/abertis/abertis1/20140722T145704-03-140604230.ts HTTP
17:19:13.029713000	192.168.1.182	8.254.202.126	HTTP	441	GET /abertis/abertis1/20140722T145704-03-140604230.ts HTTP/1.1
17:19:13.042094000	192.168.1.192	192.168.1.182	HTTP	474	GET http://abertis-test.hls.adaptive.level3.net/abertis/abertis1/01.m3u8 HTTP/1.1
17:19:13.084924000	192.168.1.182	8.254.202.126	HTTP	413	GET /abertis/abertis1/01.m3u8 HTTP/1.1
17:19:13.170717000	8.254.202.126	192.168.1.182	HTTP	763	HTTP/1.1 200 OK (application/x-mpegurl)
17:19:13.171092000	192.168.1.182	192.168.1.192	HTTP	775	HTTP/1.1 200 OK (application/x-mpegurl)
17:19:13.466229000	8.254.202.126	192.168.1.182	HTTP	516	HTTP/1.1 200 OK (video/mp2t)
17:19:13.466421000	192.168.1.182	192.168.1.192	HTTP	15128	HTTP/1.1 200 OK (video/mp2t)
17:19:13.478323000	192.168.1.192	192.168.1.182	HTTP	502	GET http://abertis-test.hls.adaptive.level3.net/abertis/abertis1/20140722T145704-03-140604231.ts HTTP
17:19:13.479037000	192.168.1.182	8.254.202.126	HTTP	441	GET /abertis/abertis1/20140722T145704-03-140604231.ts HTTP/1.1
17:19:14.238795000	8.254.202.126	192.168.1.182	HTTP	796	HTTP/1.1 200 OK (video/mp2t)
17:19:14.239053000	192.168.1.182	192.168.1.192	HTTP	21248	HTTP/1.1 200 OK (video/mp2t)
17:19:14.256578000	192.168.1.192	192.168.1.182	HTTP	474	GET http://abertis-test.hls.adaptive.level3.net/abertis/abertis1/01.m3u8 HTTP/1.1
17:19:14.257598000	192.168.1.182	8.254.202.126	HTTP	413	GET /abertis/abertis1/01.m3u8 HTTP/1.1
17:19:14.320575000	8.254.202.126	192.168.1.182	HTTP	1146	HTTP/1.1 200 OK (application/x-mpegurl)
17:19:14.321398000	192.168.1.182	192.168.1.192	HTTP	775	HTTP/1.1 200 OK (application/x-mpegurl)
17:19:14.394184000	192.168.1.192	192.168.1.182	HTTP	502	GET http://abertis-test.hls.adaptive.level3.net/abertis/abertis1/20140722T145704-01-140604231.ts HTTP
17:19:14.394899000	192.168.1.182	8.254.202.126	HTTP	441	GET /abertis/abertis1/20140722T145704-01-140604231.ts HTTP/1.1
17:19:15.314788000	8.254.202.126	192.168.1.182	HTTP	1022	HTTP/1.1 200 OK (video/mp2t)
17:19:15.315152000	192.168.1.182	192.168.1.192	HTTP	1846	HTTP/1.1 200 OK (video/mp2t)
17:19:15.333609000	192.168.1.192	192.168.1.182	HTTP	502	GET http://abertis-test.hls.adaptive.level3.net/abertis/abertis1/20140722T145704-01-140604232.ts HTTP

Figura 39. Captura 2 HLS propiedad de pruebas

Vemos que se piden varios archivos de vídeo de manera consecutiva al recuperar la conexión y se produce un cambio de calidad, para ello simplemente se solicita un manifest de una calidad inferior y se comienzan a solicitar los .ts de esa calidad.

Al realizar un corte de 30 segundos, desde las 17:22:00 a las 17:22:30, y tras realizar varias pruebas, la reproducción se ha parado y ya no ha sido capaz de continuar, teniendo que reanudar la misma de forma manual.



Time	Source	Destination	Protocol	Length	Info
17:21:59.150604000	192.168.1.182	192.168.1.192	HTTP	30170	HTTP/1.1 200 OK (video/mp2t)
17:22:31.695406000	192.168.1.192	192.168.1.182	HTTP	474	GET http://abertis-test.hls.adaptive.level3.net/abertis/abertis1/02.m3u8 HTTP/1.1
17:22:31.696352000	192.168.1.182	8.254.202.254	HTTP	413	GET /abertis/abertis1/02.m3u8 HTTP/1.1
17:22:31.782160000	192.168.1.182	8.254.202.254	HTTP	413	GET /abertis/abertis1/02.m3u8 HTTP/1.1
17:22:31.865617000	8.254.202.254	192.168.1.182	HTTP	763	HTTP/1.1 200 OK (application/x-mpegurl)
17:22:31.866593000	192.168.1.182	192.168.1.192	HTTP	775	HTTP/1.1 200 OK (application/x-mpegurl)
17:22:31.876138000	192.168.1.192	192.168.1.182	HTTP	502	GET http://abertis-test.hls.adaptive.level3.net/abertis/abertis1/20140722T145704-02-140604243.ts HTTP
17:22:31.876813000	192.168.1.182	8.254.202.254	HTTP	441	GET /abertis/abertis1/20140722T145704-02-140604243.ts HTTP/1.1
17:22:32.117329000	8.254.202.254	192.168.1.182	HTTP	550	HTTP/1.1 200 OK (video/mp2t)
17:22:32.117587000	192.168.1.182	192.168.1.192	HTTP	21002	HTTP/1.1 200 OK (video/mp2t)
17:22:32.148700000	192.168.1.192	192.168.1.182	HTTP	502	GET http://abertis-test.hls.adaptive.level3.net/abertis/abertis1/20140722T145704-02-140604244.ts HTTP
17:22:32.319940000	8.254.202.254	192.168.1.182	HTTP	628	HTTP/1.1 200 OK (video/mp2t)
17:22:32.320129000	192.168.1.182	192.168.1.192	HTTP	9400	HTTP/1.1 200 OK (video/mp2t)
17:22:32.334966000	192.168.1.192	192.168.1.182	HTTP	502	GET http://abertis-test.hls.adaptive.level3.net/abertis/abertis1/20140722T145704-02-140604245.ts HTTP
17:22:32.335655000	192.168.1.182	8.254.202.254	HTTP	441	GET /abertis/abertis1/20140722T145704-02-140604245.ts HTTP/1.1
17:22:32.613139000	8.254.202.254	192.168.1.182	HTTP	904	HTTP/1.1 200 OK (video/mp2t)
17:22:32.613316000	192.168.1.182	192.168.1.192	HTTP	8216	HTTP/1.1 200 OK (video/mp2t)
17:22:32.623547000	192.168.1.192	192.168.1.182	HTTP	502	GET http://abertis-test.hls.adaptive.level3.net/abertis/abertis1/20140722T145704-02-140604246.ts HTTP
17:22:32.624252000	192.168.1.182	8.254.202.254	HTTP	441	GET /abertis/abertis1/20140722T145704-02-140604246.ts HTTP/1.1
17:22:32.860743000	8.254.202.254	192.168.1.182	HTTP	270	HTTP/1.1 200 OK (video/mp2t)
17:22:32.861184000	192.168.1.182	192.168.1.192	HTTP	23642	HTTP/1.1 200 OK (video/mp2t)
17:22:32.883436000	192.168.1.192	192.168.1.182	HTTP	502	GET http://abertis-test.hls.adaptive.level3.net/abertis/abertis1/20140722T145704-02-140604247.ts HTTP
17:22:32.884367000	192.168.1.182	8.254.202.254	HTTP	441	GET /abertis/abertis1/20140722T145704-02-140604247.ts HTTP/1.1
17:22:33.044328000	8.254.202.254	192.168.1.182	HTTP	248	HTTP/1.1 200 OK (video/mp2t)
17:22:33.044667000	192.168.1.182	192.168.1.192	HTTP	32380	HTTP/1.1 200 OK (video/mp2t)
17:22:33.059493000	192.168.1.192	192.168.1.182	HTTP	502	GET http://abertis-test.hls.adaptive.level3.net/abertis/abertis1/20140722T145704-02-140604248.ts HTTP
17:22:33.060132000	192.168.1.182	8.254.202.254	HTTP	441	GET /abertis/abertis1/20140722T145704-02-140604248.ts HTTP/1.1
17:22:33.294605000	8.254.202.254	192.168.1.182	HTTP	580	HTTP/1.1 200 OK (video/mp2t)
17:22:33.294971000	192.168.1.182	192.168.1.192	HTTP	19572	HTTP/1.1 200 OK (video/mp2t)
17:22:33.313464000	192.168.1.192	192.168.1.182	HTTP	502	GET http://abertis-test.hls.adaptive.level3.net/abertis/abertis1/20140722T145704-02-140604249.ts HTTP

Figura 40. Captura 3 HLS propiedad de pruebas

En la captura de tráfico que realizamos se puede ver como el iPad solicita segmentos de audio (calidad 02) a la CDN y ésta responde; pero lo cierto es que el reproductor no reanuda. Finalmente, el iPad se queda solicitando un Manifest al que la CDN ya no responde.

3.3.1.4. Pruebas HLS cambiando la duración del segmento

Vamos a realizar una batería de pruebas variando la duración de los segmentos con el protocolo HLS (posteriormente lo haremos con los demás) para analizar el comportamiento y poder compararlo con las pruebas realizadas anteriormente que emplean los valores estándar.

En el caso de HLS vamos a utilizar la misma propiedad que empleamos anteriormente reproduciendo en un player a través del navegador de Safari. Empleamos únicamente este player ya que para las propiedades que hemos empleado en primera instancia de los proveedores de canales de TV, como Antena 3 u Ono, no podemos cambiar los parámetros de configuración.

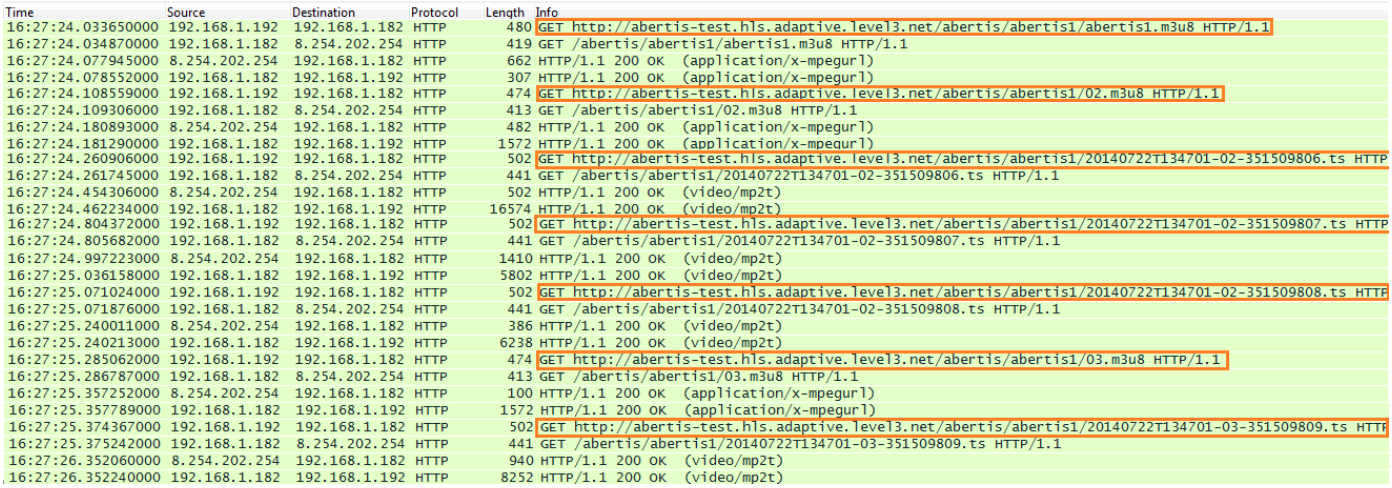
En HLS la duración habitual del segmento es de 8/10 segundos. En este caso vamos a variar este parámetro configurando 4 segundos como duración del segmento. Para ello accedemos al codificador y cambiamos este parámetro de configuración.

La propiedad empleada tiene 2 calidades de vídeo y una de audio. Si accedemos al sub-manifest podemos ver que la duración del segmento ha cambiado a 4 segundos y ahora tenemos 30 segmentos (.ts) dentro del sub-manifest. Esto es así porque tenemos que tener tantos archivos .ts como sean necesarios para cubrir la duración del DVR, que en nuestro caso es de 120 segundos. En el caso anterior, con una duración del segmento de 10 segundos, teníamos 12 .ts dentro del sub-manifest.

Figura 41. Sub-Manifest HLS cambiando la duración del segmento

Por tanto, podemos extraer que al disminuir el tamaño del segmento, el sub-manifest ocupa más.

En primer lugar realizamos una captura al inicio de la reproducción.



Time	Source	Destination	Protocol	Length	Info
16:27:24.033650000	192.168.1.192	192.168.1.182	HTTP	480	GET http://abertis-test.hls.adaptive.level3.net/abertis/abertis1/abertis1.m3u8 HTTP/1.1
16:27:24.034870000	192.168.1.182	8.254.202.254	HTTP	419	GET /abertis/abertis1/abertis1.m3u8 HTTP/1.1
16:27:24.077945000	8.254.202.254	192.168.1.182	HTTP	662	HTTP/1.1 200 OK (application/x-mpegurl)
16:27:24.078552000	192.168.1.182	192.168.1.192	HTTP	307	HTTP/1.1 200 OK (application/x-mpegurl)
16:27:24.108559000	192.168.1.192	192.168.1.182	HTTP	474	GET http://abertis-test.hls.adaptive.level3.net/abertis/abertis1/02.m3u8 HTTP/1.1
16:27:24.109306000	192.168.1.182	8.254.202.254	HTTP	413	GET /abertis/abertis1/02.m3u8 HTTP/1.1
16:27:24.180893000	8.254.202.254	192.168.1.182	HTTP	482	HTTP/1.1 200 OK (application/x-mpegurl)
16:27:24.181290000	192.168.1.182	192.168.1.192	HTTP	1572	HTTP/1.1 200 OK (application/x-mpegurl)
16:27:24.260906000	192.168.1.192	192.168.1.182	HTTP	502	GET http://abertis-test.hls.adaptive.level3.net/abertis/abertis1/20140722T134701-02-351509806.ts HTTP
16:27:24.261745000	192.168.1.182	8.254.202.254	HTTP	441	GET /abertis/abertis1/20140722T134701-02-351509806.ts HTTP/1.1
16:27:24.454306000	8.254.202.254	192.168.1.182	HTTP	502	HTTP/1.1 200 OK (video/mp2t)
16:27:24.462234000	192.168.1.182	192.168.1.192	HTTP	16574	HTTP/1.1 200 OK (video/mp2t)
16:27:24.804372000	192.168.1.192	192.168.1.182	HTTP	502	GET http://abertis-test.hls.adaptive.level3.net/abertis/abertis1/20140722T134701-02-351509807.ts HTTP
16:27:24.805682000	192.168.1.182	8.254.202.254	HTTP	441	GET /abertis/abertis1/20140722T134701-02-351509807.ts HTTP/1.1
16:27:24.997223000	8.254.202.254	192.168.1.182	HTTP	1410	HTTP/1.1 200 OK (video/mp2t)
16:27:25.036158000	192.168.1.182	192.168.1.192	HTTP	5802	HTTP/1.1 200 OK (video/mp2t)
16:27:25.071024000	192.168.1.192	192.168.1.182	HTTP	502	GET http://abertis-test.hls.adaptive.level3.net/abertis/abertis1/20140722T134701-02-351509808.ts HTTP
16:27:25.071876000	192.168.1.182	8.254.202.254	HTTP	441	GET /abertis/abertis1/20140722T134701-02-351509808.ts HTTP/1.1
16:27:25.240011000	8.254.202.254	192.168.1.182	HTTP	386	HTTP/1.1 200 OK (video/mp2t)
16:27:25.240213000	192.168.1.182	192.168.1.192	HTTP	6238	HTTP/1.1 200 OK (video/mp2t)
16:27:25.285062000	192.168.1.192	192.168.1.182	HTTP	474	GET http://abertis-test.hls.adaptive.level3.net/abertis/abertis1/03.m3u8 HTTP/1.1
16:27:25.286787000	192.168.1.182	8.254.202.254	HTTP	413	GET /abertis/abertis1/03.m3u8 HTTP/1.1
16:27:25.357252000	8.254.202.254	192.168.1.182	HTTP	100	HTTP/1.1 200 OK (application/x-mpegurl)
16:27:25.357789000	192.168.1.182	192.168.1.192	HTTP	1572	HTTP/1.1 200 OK (application/x-mpegurl)
16:27:25.374367000	192.168.1.192	192.168.1.182	HTTP	502	GET http://abertis-test.hls.adaptive.level3.net/abertis/abertis1/20140722T134701-03-351509809.ts HTTP
16:27:25.375242000	192.168.1.182	8.254.202.254	HTTP	441	GET /abertis/abertis1/20140722T134701-03-351509809.ts HTTP/1.1
16:27:26.352060000	8.254.202.254	192.168.1.182	HTTP	940	HTTP/1.1 200 OK (video/mp2t)
16:27:26.352240000	192.168.1.182	192.168.1.192	HTTP	8252	HTTP/1.1 200 OK (video/mp2t)

Figura 42. Captura del inicio de HLS cambiando la duración del segmento

Para empezar se pide el Manifest (abertis1.m3u8) y a continuación el sub-manifest correspondiente a la calidad más baja (02.m3u8). En este caso la correspondiente al audio. Se comienzan a transmitir segmentos (.ts). En una situación estable los segmentos se piden cada 4 segundos (duración establecida del segmento). Vemos que en un determinado momento (en el que las condiciones lo permiten) hay un cambio de calidad, en este caso, a la del stream 03 (segunda calidad más alta).

Siguiendo la metodología habitual, en primer lugar provocamos un corte prolongado para ver cuántos segmentos hay almacenados en el buffer. El vídeo se reproduce durante 12 segundos aproximadamente antes de cortarse, lo que equivale a 3 segmentos de duración 4 segundos.

A continuación introducimos cortes aumentando progresivamente el tiempo. Al provocar un corte de 4 segundos, desde las 16:30:10 hasta las 16:30:14, la reproducción no se para.

ClientBeginR...	ServerDone...	Result	Protocol	Host	URL
16:30:03.893	16:30:03.963	200	HTTP	abertis-test.hls.adaptive.level3.net	/abertis/abertis1/01.m3u8
16:30:03.974	16:30:04.532	200	HTTP	abertis-test.hls.adaptive.level3.net	/abertis/abertis1/20140722T134701-01-351509849.ts
16:30:07.905	16:30:07.976	200	HTTP	abertis-test.hls.adaptive.level3.net	/abertis/abertis1/01.m3u8
16:30:09.915	16:30:09.986	200	HTTP	abertis-test.hls.adaptive.level3.net	/abertis/abertis1/01.m3u8
16:30:10.001	16:30:19.405	200	HTTP	abertis-test.hls.adaptive.level3.net	/abertis/abertis1/20140722T134701-01-351509850.ts
16:30:15.948	16:30:16.081	200	HTTP	abertis-test.hls.adaptive.level3.net	/abertis/abertis1/03.m3u8
16:30:16.090	16:30:16.475	200	HTTP	abertis-test.hls.adaptive.level3.net	/abertis/abertis1/20140722T134701-03-351509849.ts
16:30:16.658	16:30:16.881	200	HTTP	abertis-test.hls.adaptive.level3.net	/abertis/abertis1/20140722T134701-03-351509850.ts
16:30:16.955	16:30:17.125	200	HTTP	abertis-test.hls.adaptive.level3.net	/abertis/abertis1/20140722T134701-03-351509851.ts
16:30:17.170	16:30:17.488	200	HTTP	abertis-test.hls.adaptive.level3.net	/abertis/abertis1/20140722T134701-03-351509852.ts
16:30:18.024	16:30:18.117	200	HTTP	abertis-test.hls.adaptive.level3.net	/abertis/abertis1/03.m3u8
16:30:20.065	16:30:20.141	200	HTTP	abertis-test.hls.adaptive.level3.net	/abertis/abertis1/03.m3u8
16:30:20.161	16:30:20.482	200	HTTP	abertis-test.hls.adaptive.level3.net	/abertis/abertis1/20140722T134701-03-351509853.ts
16:30:24.056	16:30:24.145	200	HTTP	abertis-test.hls.adaptive.level3.net	/abertis/abertis1/03.m3u8
16:30:24.155	16:30:24.468	200	HTTP	abertis-test.hls.adaptive.level3.net	/abertis/abertis1/20140722T134701-03-351509854.ts
16:30:24.493	16:30:24.571	200	HTTP	abertis-test.hls.adaptive.level3.net	/abertis/abertis1/01.m3u8
16:30:24.580	16:30:24.828	200	HTTP	abertis-test.hls.adaptive.level3.net	/abertis/abertis1/20140722T134701-01-351509853.ts
16:30:24.853	16:30:25.132	200	HTTP	abertis-test.hls.adaptive.level3.net	/abertis/abertis1/20140722T134701-01-351509854.ts
16:30:28.528	16:30:28.665	200	HTTP	abertis-test.hls.adaptive.level3.net	/abertis/abertis1/01.m3u8
16:30:28.674	16:30:29.124	200	HTTP	abertis-test.hls.adaptive.level3.net	/abertis/abertis1/20140722T134701-01-351509855.ts
16:30:32.543	16:30:32.618	200	HTTP	abertis-test.hls.adaptive.level3.net	/abertis/abertis1/01.m3u8
16:30:32.629	16:30:33.391	200	HTTP	abertis-test.hls.adaptive.level3.net	/abertis/abertis1/20140722T134701-01-351509856.ts

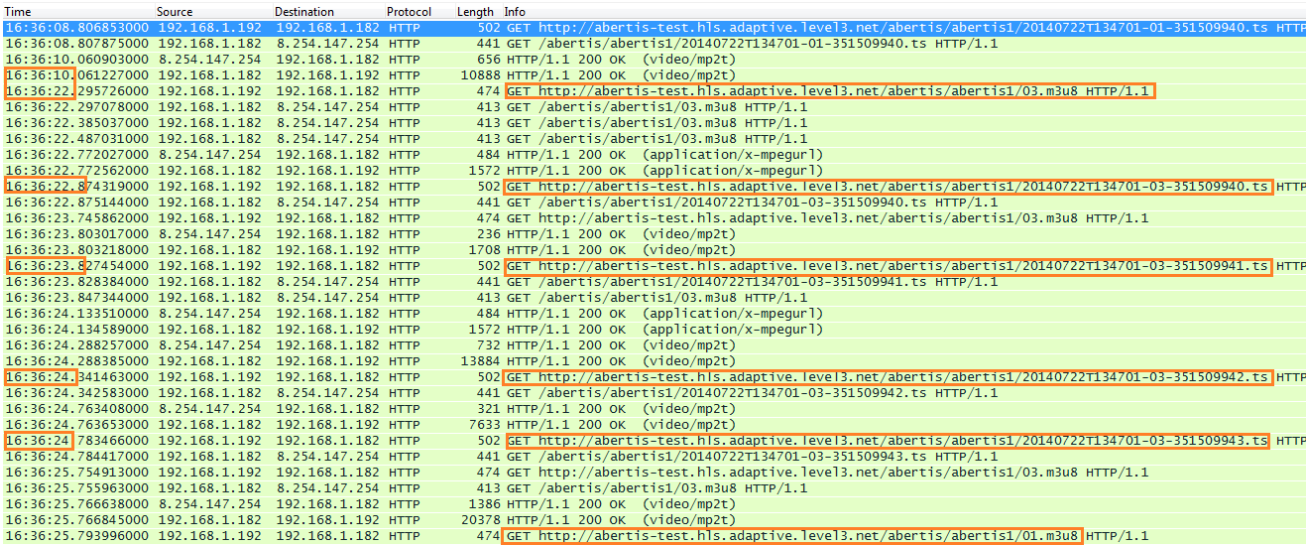
Figura 43. Captura 1 HLS propiedad de pruebas cambiando la duración del segmento

Podemos ver como antes del corte se está reproduciendo en la calidad más alta (01); al finalizar el corte hay un cambio de calidad a la más baja de vídeo (03) y poco después se vuelve a recuperar la calidad más alta.

Mediante la captura mostrada anteriormente con el programa Fiddler sólo aparecen las peticiones realizadas a la CDN, pero con el Wireshark podemos ver las respuestas como hemos mostrados en casos anteriores. Siempre se realiza una descarga del sub-manifest por cada archivo .ts (aunque con Fiddler no aparezca así).

Al provocar un corte de 8 segundos, la reproducción se cortó durante 1 segundo. Dependiendo del momento en el que se inicie el corte podría cortarse o no, ya que hay que tener en cuenta que el tiempo de vídeo guardado en el buffer no tiene siempre un valor fijo.

Al realizar un corte de 12 segundos, desde las 16:36:10 a las 16:36:22, la reproducción se para durante unos 5/6 segundos y posteriormente continúa. Hemos podido apreciar que al introducir estos cortes, se realizan bastantes cambios de calidades.



Time	Source	Destination	Protocol	Length	Info
16:36:08.806853000	192.168.1.192	192.168.1.182	HTTP	502	GET http://abertis-test.hls.adaptive.level3.net/abertis/abertis1/20140722T134701-01-351509940.ts HTTP
16:36:08.807875000	192.168.1.182	8.254.147.254	HTTP	441	GET /abertis/abertis1/20140722T134701-01-351509940.ts HTTP/1.1
16:36:10.060903000	8.254.147.254	192.168.1.182	HTTP	656	HTTP/1.1 200 OK (video/mp2t)
16:36:10.061227000	192.168.1.182	192.168.1.192	HTTP	10888	HTTP/1.1 200 OK (video/mp2t)
16:36:22.295726000	192.168.1.192	192.168.1.182	HTTP	474	GET http://abertis-test.hls.adaptive.level3.net/abertis/abertis1/03.m3u8 HTTP/1.1
16:36:22.297078000	192.168.1.182	8.254.147.254	HTTP	413	GET /abertis/abertis1/03.m3u8 HTTP/1.1
16:36:22.385037000	192.168.1.182	8.254.147.254	HTTP	413	GET /abertis/abertis1/03.m3u8 HTTP/1.1
16:36:22.487031000	192.168.1.182	8.254.147.254	HTTP	413	GET /abertis/abertis1/03.m3u8 HTTP/1.1
16:36:22.772027000	8.254.147.254	192.168.1.182	HTTP	484	HTTP/1.1 200 OK (application/x-mpegurl)
16:36:22.772562000	192.168.1.182	192.168.1.192	HTTP	1572	HTTP/1.1 200 OK (application/x-mpegurl)
16:36:22.874319000	192.168.1.192	192.168.1.182	HTTP	502	GET http://abertis-test.hls.adaptive.level3.net/abertis/abertis1/20140722T134701-03-351509940.ts HTTP
16:36:22.875144000	192.168.1.182	8.254.147.254	HTTP	441	GET /abertis/abertis1/20140722T134701-03-351509940.ts HTTP/1.1
16:36:23.745862000	192.168.1.192	192.168.1.182	HTTP	474	GET http://abertis-test.hls.adaptive.level3.net/abertis/abertis1/03.m3u8 HTTP/1.1
16:36:23.803017000	8.254.147.254	192.168.1.182	HTTP	236	HTTP/1.1 200 OK (video/mp2t)
16:36:23.803218000	192.168.1.182	192.168.1.192	HTTP	1708	HTTP/1.1 200 OK (video/mp2t)
16:36:23.827454000	192.168.1.192	192.168.1.182	HTTP	502	GET http://abertis-test.hls.adaptive.level3.net/abertis/abertis1/20140722T134701-03-351509941.ts HTTP
16:36:23.828384000	192.168.1.182	8.254.147.254	HTTP	441	GET /abertis/abertis1/20140722T134701-03-351509941.ts HTTP/1.1
16:36:23.847344000	192.168.1.182	8.254.147.254	HTTP	413	GET /abertis/abertis1/03.m3u8 HTTP/1.1
16:36:24.133510000	8.254.147.254	192.168.1.182	HTTP	484	HTTP/1.1 200 OK (application/x-mpegurl)
16:36:24.134589000	192.168.1.182	192.168.1.192	HTTP	1572	HTTP/1.1 200 OK (application/x-mpegurl)
16:36:24.288257000	8.254.147.254	192.168.1.182	HTTP	732	HTTP/1.1 200 OK (video/mp2t)
16:36:24.288385000	192.168.1.182	192.168.1.192	HTTP	13884	HTTP/1.1 200 OK (video/mp2t)
16:36:24.341463000	192.168.1.192	192.168.1.182	HTTP	502	GET http://abertis-test.hls.adaptive.level3.net/abertis/abertis1/20140722T134701-03-351509942.ts HTTP
16:36:24.342583000	192.168.1.182	8.254.147.254	HTTP	441	GET /abertis/abertis1/20140722T134701-03-351509942.ts HTTP/1.1
16:36:24.763408000	8.254.147.254	192.168.1.182	HTTP	321	HTTP/1.1 200 OK (video/mp2t)
16:36:24.763653000	192.168.1.182	192.168.1.192	HTTP	7633	HTTP/1.1 200 OK (video/mp2t)
16:36:24.783466000	192.168.1.192	192.168.1.182	HTTP	502	GET http://abertis-test.hls.adaptive.level3.net/abertis/abertis1/20140722T134701-03-351509943.ts HTTP
16:36:24.784417000	192.168.1.182	8.254.147.254	HTTP	441	GET /abertis/abertis1/20140722T134701-03-351509943.ts HTTP/1.1
16:36:25.754913000	192.168.1.192	192.168.1.182	HTTP	474	GET http://abertis-test.hls.adaptive.level3.net/abertis/abertis1/03.m3u8 HTTP/1.1
16:36:25.755963000	192.168.1.182	8.254.147.254	HTTP	413	GET /abertis/abertis1/03.m3u8 HTTP/1.1
16:36:25.766638000	8.254.147.254	192.168.1.182	HTTP	1386	HTTP/1.1 200 OK (video/mp2t)
16:36:25.766845000	192.168.1.182	192.168.1.192	HTTP	20378	HTTP/1.1 200 OK (video/mp2t)
16:36:25.793996000	192.168.1.192	192.168.1.182	HTTP	474	GET http://abertis-test.hls.adaptive.level3.net/abertis/abertis1/01.m3u8 HTTP/1.1

Figura 44. Captura 2 HLS propiedad de pruebas cambiando la duración del segmento

En muy pocos segundos, se solicitan un gran número de archivos .ts. En la captura que recogimos se solicitaban 6 segmentos en 4 segundos con un cambio de calidad (de la 03 a la 01).

Al provocar un corte de 16 segundos, el reproductor se paraba y ya no reanudaba el vídeo. Se quedaba solicitando el sub-manifest pero la CDN no respondía.

A continuación vemos un cuadro resumen de los resultados obtenidos con los distintos reproductores de HLS.

HLS			
PLATAFORMA/PLAYER	**TAMAÑO DEL BUFFER**	**DURACIÓN CHUNKS**	**MAXIMO CORTE SOPORTADO (CON REANUDACIÓN)**
Nubeox	15/18 segundos	8 segundos	20/24 segundos
Ono	20/30 segundos	10 segundos	20/24 segundos
Safari	20 segundos	10 segundos	< 30 segundos
Safari	12 segundos	4 segundos	16 segundos

Tabla 5. Resumen pruebas HLS

Podemos determinar, que al disminuir la duración del segmento hemos provocado que el tamaño del sub-manifest crezca ya que contiene muchos más archivos .ts, y además, que se produzcan muchas más descargas del sub-manifest (una cada 4 segundos). Esto es así porque en HLS cada vez que se descarga un segmento se solicita el sub-manifest con los archivos de vídeo (.ts). Además, el buffering inicial cuando la duración del segmento es menor es más rápido, es decir, en menos tiempo se piden los segmentos necesarios para llenar el buffer. Con las pruebas realizadas también podemos ver que el

player es más robusto ante cortes empleando la duración del segmento que marca el estándar (8/10 segundos).

Respecto a los reproductores de los 2 operadores no hemos encontrado grandes diferencias entre ellos, quizás que el buffer del player de Ono tiene algo de más capacidad que el de Nubeox, lo que lo haría algo más robusto ante cortes. La robustez del reproductor de Safari ante cortes comparado con el reproductor de Ono es bastante similar.

Cuando hablamos del máximo corte soportado, estamos haciendo referencia al tiempo de corte introducido que aún provocando interrupciones en la señal de vídeo reproducida (algo no deseable), se consigue recuperar el contenido sin tener que reiniciar manualmente el player. Podemos decir que esto es una característica más propia del reproductor empleado que del protocolo utilizado en sí (aunque cada reproductor vaya asociado/ligado a un protocolo).

En general, cuando hemos empleado una duración del segmento mayor, el máximo corte soportado con reanudación posterior del player ha sido también mayor.

3.3.1.5. Pruebas HLS cambiando la duración del DVR

Otro parámetro característico de los protocolos de transmisión de vídeo adaptativo es el DVR. Como ya hemos visto anteriormente, el DVR es el contenido en tiempo que un usuario puede ver respecto al último intervalo de tiempo del live stream. Vamos a variar el DVR utilizando las propiedades para pruebas de alguno de los protocolos estudiados (Antena 3 u Ono ofrecen servicios encriptados dónde no podemos cambiar los parámetros).

A priori, un cambio del tamaño del DVR no debe afectar a la reproducción del vídeo, simplemente nos permitirá retroceder más o menos el vídeo en función de la duración configurada.

En HLS, el DVR influye principalmente en la estructura del sub-manifest (archivo .m3u8 de cada una de las calidades). Si aumentamos el valor del DVR, nos van a aparecer muchos más archivos de vídeo (.ts) en el sub-manifest puesto que tiene que haber tantos .ts como sean necesarios para cubrir la duración del DVR.

A modo de ejemplo, en la propiedad que hemos venido utilizando teníamos configurado un DVR de 120 segundos, lo que nos permite retroceder el vídeo en el player un máximo de 2 minutos. Si nos descargábamos el sub-manifest, con la duración estándar de 10 segundos, teníamos un total de 12 archivos .ts en la lista. Si por ejemplo queremos un DVR de 5 minutos, tendríamos que la playlist pasa de tener 12 archivos .ts a tener 30 archivos .ts.

Realizamos esta prueba cambiando el valor del DVR y pudimos observar como a través del reproductor podíamos retroceder el tiempo configurado en el parámetro del DVR.

Cabe destacar que en el caso de HLS, el valor del DVR era modificable de manera sencilla a través del codificador (al igual que el tamaño del segmento). En el caso de Smooth Streaming, como veremos en el apartado 3.3.2.4., hay que acceder a un administrador especial de las propiedades de la CDN para cambiar el DVR.

Por tanto, cuanto más grande sea el DVR, la playlist (en el caso de HLS) se puede ir haciendo más grande, y ya se ha comenzado a comprimir la playlist para enviarla en formato gzip [39] de una manera más rápida.

3.3.2. Pruebas Smooth Streaming

Para probar el protocolo de Smooth Streaming seguiremos la misma metodología que en HLS, en primer lugar probaremos el comportamiento sobre el reproductor de un operador de telecomunicaciones. Y posteriormente emplearemos el reproductor estándar de Silverlight con una propiedad pública.

Figura 45. Escenario HSS (Smooth Streaming)

En este caso sólo necesitamos el ordenador portátil y el dispositivo Net.Storm para introducir los cortes en la señal de vídeo.

3.3.2.1. Pruebas HSS sobre el player de Ono

Como hemos realizado con el protocolo HLS, vamos a ver el comportamiento del reproductor de Ono para Smooth Streaming. Cabe destacar que actualmente A3 con su plataforma Nubeox no dispone de ningún sistema para reproducir contenido HSS pero está buscando soluciones para poderlo implementar (a través de TV conectadas).

Para comenzar probamos cuantos fragmentos de vídeo guarda el buffer para no producirse un corte en la reproducción si tenemos una pérdida de conexión. Esta pérdida la generamos con el dispositivo de Albedo. Tras el inicio del corte el vídeo se para tras 6 segundos, teniendo en cuenta que la duración de los fragmentos es de 2 segundos, podemos concluir que tenemos hasta 3 fragmentos "futuros" en el buffer.

A continuación introducimos cortes en la señal, aumentando el tiempo de la interrupción. Si introducimos un corte de 2 segundos de duración, desde las 17:24:30 a 17:24:32, obtenemos:

Time	Source	Destination	Protocol	Length	Info
17:24:30.452201000	192.168.1.92	4.26.233.254	HTTP	424	GET /livefeed/ono/TNT.isml/QualityLevels(2500000)/Fragments(video=1048586538047543) HTTP/1.1
17:24:30.464683000	192.168.1.92	4.26.233.254	HTTP	426	GET /livefeed/ono/TNT.isml/QualityLevels(96000)/Fragments(audio_und=1048586540205210) HTTP/1.1
17:24:34.233958000	4.26.233.254	192.168.1.92	HTTP	1412	HTTP/1.1 200 OK (video/mp4)
17:24:34.248818000	192.168.1.92	4.26.233.254	HTTP	426	GET /livefeed/ono/TNT.isml/QualityLevels(96000)/Fragments(audio_und=1048586560045210) HTTP/1.1
17:24:34.381909000	4.26.233.254	192.168.1.92	HTTP	638	HTTP/1.1 200 OK (video/mp4)
17:24:34.461417000	192.168.1.92	4.26.233.254	HTTP	426	GET /livefeed/ono/TNT.isml/QualityLevels(96000)/Fragments(audio_und=1048586580098543) HTTP/1.1
17:24:34.544361000	4.26.233.254	192.168.1.92	HTTP	285	HTTP/1.1 200 OK (video/mp4)
17:24:39.530660000	4.26.233.254	192.168.1.92	HTTP	86	HTTP/1.1 200 OK (video/mp4)
17:24:39.548904000	192.168.1.92	4.26.233.254	HTTP	426	GET /livefeed/ono/TNT.isml/QualityLevels(96000)/Fragments(audio_und=1048586600151877) HTTP/1.1
17:24:39.565381000	192.168.1.92	4.26.233.254	HTTP	423	GET /livefeed/ono/TNT.isml/QualityLevels(300000)/Fragments(video=1048586558047543) HTTP/1.1
17:24:39.614821000	4.26.233.254	192.168.1.92	HTTP	1033	HTTP/1.1 200 OK (video/mp4)
17:24:39.703595000	4.26.233.254	192.168.1.92	HTTP	445	HTTP/1.1 200 OK (video/mp4)
17:24:39.727248000	192.168.1.92	4.26.233.254	HTTP	423	GET /livefeed/ono/TNT.isml/QualityLevels(300000)/Fragments(video=1048586578047543) HTTP/1.1
17:24:39.738688000	192.168.1.92	4.26.233.254	HTTP	426	GET /livefeed/ono/TNT.isml/QualityLevels(96000)/Fragments(audio_und=1048586620205210) HTTP/1.1
17:24:39.813528000	4.26.233.254	192.168.1.92	HTTP	1011	HTTP/1.1 200 OK (video/mp4)
17:24:39.861101000	4.26.233.254	192.168.1.92	HTTP	497	HTTP/1.1 200 OK (video/mp4)
17:24:39.917032000	192.168.1.92	4.26.233.254	HTTP	424	GET /livefeed/ono/TNT.isml/QualityLevels(2500000)/Fragments(video=1048586598047543) HTTP/1.1
17:24:40.443380000	192.168.1.92	4.26.233.254	HTTP	426	GET /livefeed/ono/TNT.isml/QualityLevels(96000)/Fragments(audio_und=1048586640045210) HTTP/1.1
17:24:40.501504000	4.26.233.254	192.168.1.92	HTTP	347	HTTP/1.1 200 OK (video/mp4)
17:24:40.615805000	4.26.233.254	192.168.1.92	HTTP	1193	HTTP/1.1 200 OK (video/mp4)

Figura 46. Captura 1 Smooth Streaming para un canal de Ono

Cabe destacar que el canal elegido tiene 4 calidades como podemos comprobar accediendo al Manifest, la más alta presenta un bit-rate de 2.500 kbps. Al realizar un corte de tan sólo 2 segundos, la reproducción no se corta, simplemente apreciamos un pequeño cambio de calidad (a la tercera) pero observamos que rápidamente se vuelve a solicitar fragmentos de la calidad más alta.

Si introducimos un corte superior a 6 segundos ya vemos como el reproductor se queda haciendo buffering y hay un corte en el vídeo que estamos reproduciendo, aunque cuando se reanuda la conexión, el player si es capaz de continuar con la reproducción.

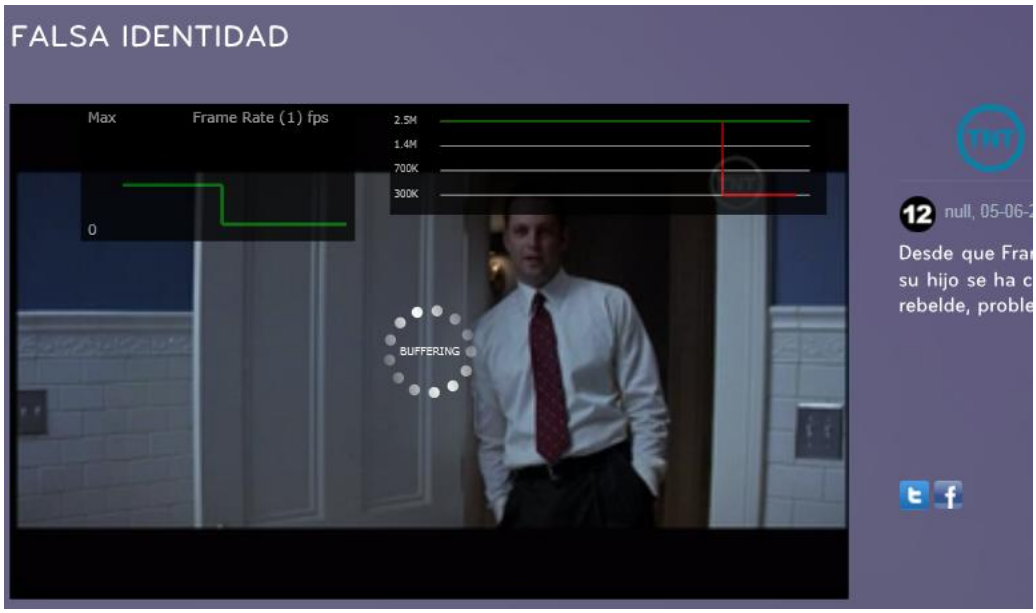
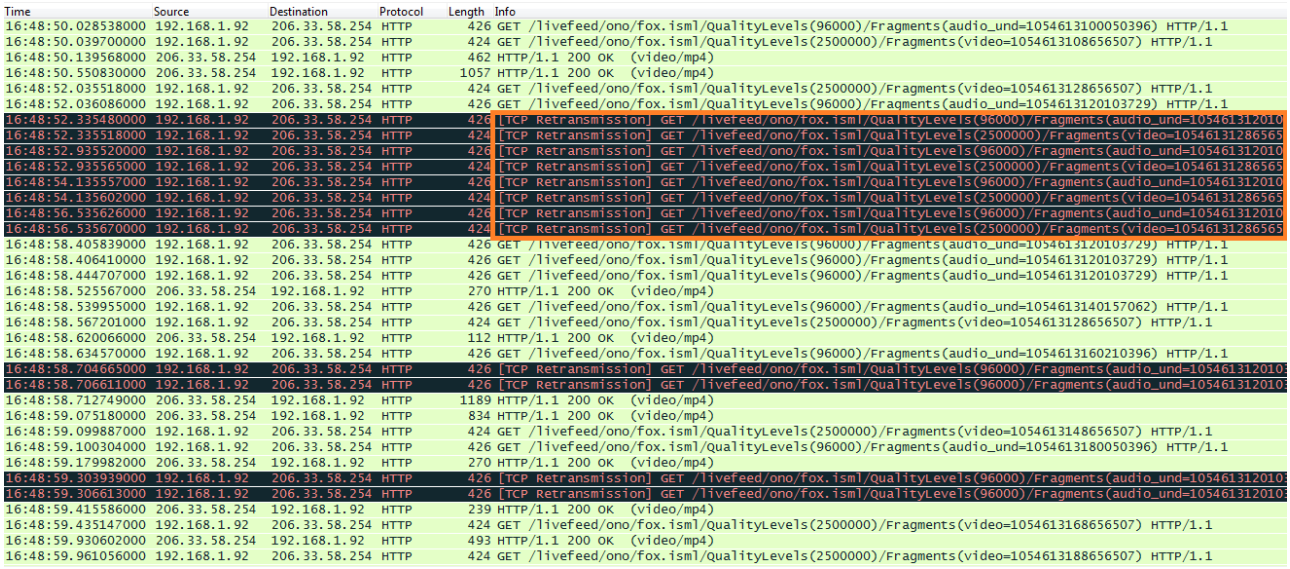


Figura 47. Reproductor Plataforma Ono (Smooth Streaming)

Mostraremos algunos ejemplos de las capturas realizadas.

Realizamos un corte de 6 segundos desde las 16:48:52 a las 16:48:58. En este caso no se produce ningún corte en la reproducción del vídeo.



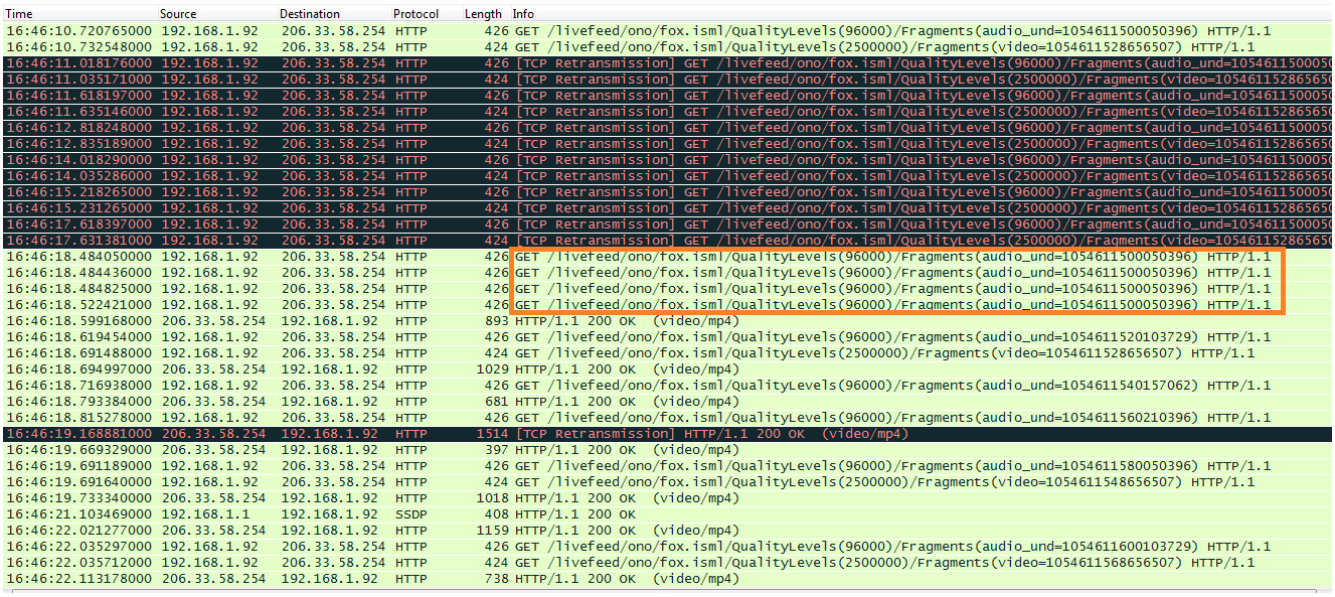
Time	Source	Destination	Protocol	Length	Info
16:48:50.028538000	192.168.1.92	206.33.58.254	HTTP	426	GET /livefeed/ono/fox.isml/QualityLevels(96000)/Fragments(audio_und=1054613100050396) HTTP/1.1
16:48:50.039700000	192.168.1.92	206.33.58.254	HTTP	424	GET /livefeed/ono/fox.isml/QualityLevels(2500000)/Fragments(video=1054613108656507) HTTP/1.1
16:48:50.139568000	206.33.58.254	192.168.1.92	HTTP	462	HTTP/1.1 200 OK (video/mp4)
16:48:50.550830000	206.33.58.254	192.168.1.92	HTTP	1057	HTTP/1.1 200 OK (video/mp4)
16:48:52.035518000	192.168.1.92	206.33.58.254	HTTP	424	GET /livefeed/ono/fox.isml/QualityLevels(2500000)/Fragments(video=1054613128656507) HTTP/1.1
16:48:52.036086000	192.168.1.92	206.33.58.254	HTTP	426	GET /livefeed/ono/fox.isml/QualityLevels(96000)/Fragments(audio_und=1054613120103729) HTTP/1.1
16:48:52.335480000	192.168.1.92	206.33.58.254	HTTP	426	[TCP Retransmission] GET /livefeed/ono/fox.isml/QualityLevels(96000)/Fragments(audio_und=10546131201
16:48:52.335518000	192.168.1.92	206.33.58.254	HTTP	424	[TCP Retransmission] GET /livefeed/ono/fox.isml/QualityLevels(2500000)/Fragments(video=10546131286565
16:48:52.935520000	192.168.1.92	206.33.58.254	HTTP	426	[TCP Retransmission] GET /livefeed/ono/fox.isml/QualityLevels(96000)/Fragments(audio_und=10546131201
16:48:52.935565000	192.168.1.92	206.33.58.254	HTTP	424	[TCP Retransmission] GET /livefeed/ono/fox.isml/QualityLevels(2500000)/Fragments(video=10546131286565
16:48:54.135557000	192.168.1.92	206.33.58.254	HTTP	426	[TCP Retransmission] GET /livefeed/ono/fox.isml/QualityLevels(96000)/Fragments(audio_und=10546131201
16:48:54.135602000	192.168.1.92	206.33.58.254	HTTP	424	[TCP Retransmission] GET /livefeed/ono/fox.isml/QualityLevels(2500000)/Fragments(video=10546131286565
16:48:56.535626000	192.168.1.92	206.33.58.254	HTTP	426	[TCP Retransmission] GET /livefeed/ono/fox.isml/QualityLevels(96000)/Fragments(audio_und=10546131201
16:48:56.535670000	192.168.1.92	206.33.58.254	HTTP	424	[TCP Retransmission] GET /livefeed/ono/fox.isml/QualityLevels(2500000)/Fragments(video=10546131286565
16:48:58.405839000	192.168.1.92	206.33.58.254	HTTP	426	GET /livefeed/ono/fox.isml/QualityLevels(96000)/Fragments(audio_und=1054613120103729) HTTP/1.1
16:48:58.406410000	192.168.1.92	206.33.58.254	HTTP	426	GET /livefeed/ono/fox.isml/QualityLevels(96000)/Fragments(audio_und=1054613120103729) HTTP/1.1
16:48:58.444707000	192.168.1.92	206.33.58.254	HTTP	426	GET /livefeed/ono/fox.isml/QualityLevels(96000)/Fragments(audio_und=1054613120103729) HTTP/1.1
16:48:58.525567000	206.33.58.254	192.168.1.92	HTTP	270	HTTP/1.1 200 OK (video/mp4)
16:48:58.539955000	192.168.1.92	206.33.58.254	HTTP	426	GET /livefeed/ono/fox.isml/QualityLevels(96000)/Fragments(audio_und=1054613140157062) HTTP/1.1
16:48:58.567201000	192.168.1.92	206.33.58.254	HTTP	424	GET /livefeed/ono/fox.isml/QualityLevels(2500000)/Fragments(video=1054613128656507) HTTP/1.1
16:48:58.620066000	206.33.58.254	192.168.1.92	HTTP	112	HTTP/1.1 200 OK (video/mp4)
16:48:58.634570000	192.168.1.92	206.33.58.254	HTTP	426	GET /livefeed/ono/fox.isml/QualityLevels(96000)/Fragments(audio_und=1054613160210396) HTTP/1.1
16:48:58.704665000	192.168.1.92	206.33.58.254	HTTP	426	[TCP Retransmission] GET /livefeed/ono/fox.isml/QualityLevels(96000)/Fragments(audio_und=10546131201
16:48:58.706611000	192.168.1.92	206.33.58.254	HTTP	426	[TCP Retransmission] GET /livefeed/ono/fox.isml/QualityLevels(96000)/Fragments(audio_und=10546131201
16:48:58.712749000	206.33.58.254	192.168.1.92	HTTP	1189	HTTP/1.1 200 OK (video/mp4)
16:48:59.075180000	206.33.58.254	192.168.1.92	HTTP	834	HTTP/1.1 200 OK (video/mp4)
16:48:59.099887000	192.168.1.92	206.33.58.254	HTTP	424	GET /livefeed/ono/fox.isml/QualityLevels(2500000)/Fragments(video=1054613148656507) HTTP/1.1
16:48:59.100304000	192.168.1.92	206.33.58.254	HTTP	426	GET /livefeed/ono/fox.isml/QualityLevels(96000)/Fragments(audio_und=1054613180050396) HTTP/1.1
16:48:59.179982000	206.33.58.254	192.168.1.92	HTTP	270	HTTP/1.1 200 OK (video/mp4)
16:48:59.303939000	192.168.1.92	206.33.58.254	HTTP	426	[TCP Retransmission] GET /livefeed/ono/fox.isml/QualityLevels(96000)/Fragments(audio_und=10546131201
16:48:59.306613000	192.168.1.92	206.33.58.254	HTTP	426	[TCP Retransmission] GET /livefeed/ono/fox.isml/QualityLevels(96000)/Fragments(audio_und=10546131201
16:48:59.415586000	206.33.58.254	192.168.1.92	HTTP	239	HTTP/1.1 200 OK (video/mp4)
16:48:59.435147000	192.168.1.92	206.33.58.254	HTTP	424	GET /livefeed/ono/fox.isml/QualityLevels(2500000)/Fragments(video=1054613168656507) HTTP/1.1
16:48:59.930602000	206.33.58.254	192.168.1.92	HTTP	493	HTTP/1.1 200 OK (video/mp4)
16:48:59.961056000	192.168.1.92	206.33.58.254	HTTP	424	GET /livefeed/ono/fox.isml/QualityLevels(2500000)/Fragments(video=1054613188656507) HTTP/1.1

Figura 48. Captura 2 Smooth Streaming para un canal de Ono

Lo que se aprecia en definitiva en las capturas realizadas por Wireshark, es la retransmisión de las peticiones del player de los fragmentos de audio y vídeo que no termina de mandar la CDN.

Poco a poco se recupera la conexión, y se vuelven a solicitar y recibir los fragmentos de audio y vídeo de manera correcta.

Al realizar un corte de 8 segundos, desde las 16:46:10 a las 16:46:18, la captura es la siguiente:



Time	Source	Destination	Protocol	Length	Info
16:46:10.720765000	192.168.1.92	206.33.58.254	HTTP	426	GET /livefeed/ono/fox.isml/QualityLevels(96000)/Fragments(audio_und=1054611500050396) HTTP/1.1
16:46:10.732548000	192.168.1.92	206.33.58.254	HTTP	424	GET /livefeed/ono/fox.isml/QualityLevels(2500000)/Fragments(video=1054611528656507) HTTP/1.1
16:46:11.018176000	192.168.1.92	206.33.58.254	HTTP	426	[TCP Retransmission] GET /livefeed/ono/fox.isml/QualityLevels(96000)/Fragments(audio_und=105461150005
16:46:11.035171000	192.168.1.92	206.33.58.254	HTTP	424	[TCP Retransmission] GET /livefeed/ono/fox.isml/QualityLevels(2500000)/Fragments(video=10546115286565
16:46:11.618197000	192.168.1.92	206.33.58.254	HTTP	426	[TCP Retransmission] GET /livefeed/ono/fox.isml/QualityLevels(96000)/Fragments(audio_und=105461150005
16:46:11.635146000	192.168.1.92	206.33.58.254	HTTP	424	[TCP Retransmission] GET /livefeed/ono/fox.isml/QualityLevels(2500000)/Fragments(video=10546115286565
16:46:12.818248000	192.168.1.92	206.33.58.254	HTTP	426	[TCP Retransmission] GET /livefeed/ono/fox.isml/QualityLevels(96000)/Fragments(audio_und=105461150005
16:46:12.835189000	192.168.1.92	206.33.58.254	HTTP	424	[TCP Retransmission] GET /livefeed/ono/fox.isml/QualityLevels(2500000)/Fragments(video=10546115286565
16:46:14.018290000	192.168.1.92	206.33.58.254	HTTP	426	[TCP Retransmission] GET /livefeed/ono/fox.isml/QualityLevels(96000)/Fragments(audio_und=105461150005
16:46:14.035286000	192.168.1.92	206.33.58.254	HTTP	424	[TCP Retransmission] GET /livefeed/ono/fox.isml/QualityLevels(2500000)/Fragments(video=10546115286565
16:46:15.218265000	192.168.1.92	206.33.58.254	HTTP	426	[TCP Retransmission] GET /livefeed/ono/fox.isml/QualityLevels(96000)/Fragments(audio_und=105461150005
16:46:15.231265000	192.168.1.92	206.33.58.254	HTTP	424	[TCP Retransmission] GET /livefeed/ono/fox.isml/QualityLevels(2500000)/Fragments(video=10546115286565
16:46:17.618397000	192.168.1.92	206.33.58.254	HTTP	426	[TCP Retransmission] GET /livefeed/ono/fox.isml/QualityLevels(96000)/Fragments(audio_und=105461150005
16:46:17.631381000	192.168.1.92	206.33.58.254	HTTP	424	[TCP Retransmission] GET /livefeed/ono/fox.isml/QualityLevels(2500000)/Fragments(video=10546115286565
16:46:18.484050000	192.168.1.92	206.33.58.254	HTTP	426	GET /livefeed/ono/fox.isml/QualityLevels(96000)/Fragments(audio_und=1054611500050396) HTTP/1.1
16:46:18.484436000	192.168.1.92	206.33.58.254	HTTP	426	GET /livefeed/ono/fox.isml/QualityLevels(96000)/Fragments(audio_und=1054611500050396) HTTP/1.1
16:46:18.484825000	192.168.1.92	206.33.58.254	HTTP	426	GET /livefeed/ono/fox.isml/QualityLevels(96000)/Fragments(audio_und=1054611500050396) HTTP/1.1
16:46:18.522421000	192.168.1.92	206.33.58.254	HTTP	426	GET /livefeed/ono/fox.isml/QualityLevels(96000)/Fragments(audio_und=1054611500050396) HTTP/1.1
16:46:18.599168000	206.33.58.254	192.168.1.92	HTTP	893	HTTP/1.1 200 OK (video/mp4)
16:46:18.619454000	192.168.1.92	206.33.58.254	HTTP	426	GET /livefeed/ono/fox.isml/QualityLevels(96000)/Fragments(audio_und=1054611520103729) HTTP/1.1
16:46:18.691488000	192.168.1.92	206.33.58.254	HTTP	424	GET /livefeed/ono/fox.isml/QualityLevels(2500000)/Fragments(video=1054611528656507) HTTP/1.1
16:46:18.694997000	206.33.58.254	192.168.1.92	HTTP	1029	HTTP/1.1 200 OK (video/mp4)
16:46:18.716938000	192.168.1.92	206.33.58.254	HTTP	426	GET /livefeed/ono/fox.isml/QualityLevels(96000)/Fragments(audio_und=1054611540157062) HTTP/1.1
16:46:18.793384000	206.33.58.254	192.168.1.92	HTTP	681	HTTP/1.1 200 OK (video/mp4)
16:46:18.815278000	192.168.1.92	206.33.58.254	HTTP	426	GET /livefeed/ono/fox.isml/QualityLevels(96000)/Fragments(audio_und=1054611560210396) HTTP/1.1
16:46:19.168881000	206.33.58.254	192.168.1.92	HTTP	1514	[TCP Retransmission] HTTP/1.1 200 OK (video/mp4)
16:46:19.669329000	206.33.58.254	192.168.1.92	HTTP	397	HTTP/1.1 200 OK (video/mp4)
16:46:19.691189000	192.168.1.92	206.33.58.254	HTTP	426	GET /livefeed/ono/fox.isml/QualityLevels(96000)/Fragments(audio_und=1054611580050396) HTTP/1.1
16:46:19.691640000	192.168.1.92	206.33.58.254	HTTP	424	GET /livefeed/ono/fox.isml/QualityLevels(2500000)/Fragments(video=1054611548656507) HTTP/1.1
16:46:19.733340000	206.33.58.254	192.168.1.92	HTTP	1018	HTTP/1.1 200 OK (video/mp4)
16:46:21.103469000	192.168.1.1	192.168.1.92	SSDP	408	HTTP/1.1 200 OK
16:46:22.021277000	206.33.58.254	192.168.1.92	HTTP	1159	HTTP/1.1 200 OK (video/mp4)
16:46:22.035297000	192.168.1.92	206.33.58.254	HTTP	426	GET /livefeed/ono/fox.isml/QualityLevels(96000)/Fragments(audio_und=1054611600103729) HTTP/1.1
16:46:22.035712000	192.168.1.92	206.33.58.254	HTTP	424	GET /livefeed/ono/fox.isml/QualityLevels(2500000)/Fragments(video=1054611568656507) HTTP/1.1
16:46:22.113178000	206.33.58.254	192.168.1.92	HTTP	738	HTTP/1.1 200 OK (video/mp4)

Figura 49. Captura 3 Smooth Streaming para un canal de Ono

En este caso el comportamiento es similar, se retransmiten segmentos de audio y vídeo, hasta que se recupera la conexión. La diferencia es que ya se aprecia un pequeño corte de aproximadamente 1 segundo.

Fuimos subiendo el tiempo con un paso de 2 segundos, y vimos como el efecto era el mismo, aumentando la duración del corte del vídeo en el player en función del tiempo total del corte. Para un corte de 20 segundos de duración, perdíamos la señal de vídeo durante unos 16 segundos aproximadamente. El límite se encontraba en 38 segundos, al realizar un corte de esta duración, el reproductor muestra un mensaje de "vídeo no disponible".

3.3.2.2. Pruebas HSS sobre el reproductor Silverlight

El siguiente paso fue realizar pruebas con el protocolo de Smooth Streaming empleando el reproductor estándar de Silverlight [2]. Como ya mencionamos, en este protocolo sólo se pide el Manifest una vez, al comienzo de la reproducción.

Utilizaremos una propiedad no encriptada que podemos visualizar con el reproductor de Silverlight. Para esta propiedad los fragmentos de Smooth Streaming tienen una duración de 2 segundos y el DVR es de 600 segundos (10 minutos). Esta propiedad presenta 4 calidades, es decir, disponemos de 4 streams con distintos bit-rates.

Figura 50. Reproductor estándar de Silverlight

El reproductor nos permite introducir la URL del Manifest y reproducir el contenido. También nos permite dar marcha atrás o avanzar en la reproducción siempre que hayamos retrocedido anteriormente. Esta característica está limitada por el DVR, como máximo podremos retroceder 10 minutos (tamaño ventana).

De nuevo y siguiendo la dinámica de las pruebas con el protocolo HLS, vamos a probar la robustez del protocolo Smooth Streaming y del player estándar.

Como vimos en la parte teórica, en Smooth Streaming sólo se pide el Manifest una vez al comienzo de la reproducción y, posteriormente, se solicitan los fragmentos de vídeo y audio. Esto es así porque en el propio Manifest viene una referencia del tiempo de inicio del vídeo, y posteriormente, en cada fragmento también hay referencias al tiempo. Los fragmentos se piden cada 2 segundos que coincide con la duración de los mismos. Esta corta duración de los fragmentos permite un buffering inicial bastante rápido comparado por ejemplo con el protocolo HLS.

A continuación podemos ver una captura del tráfico "live" de Smooth Streaming:

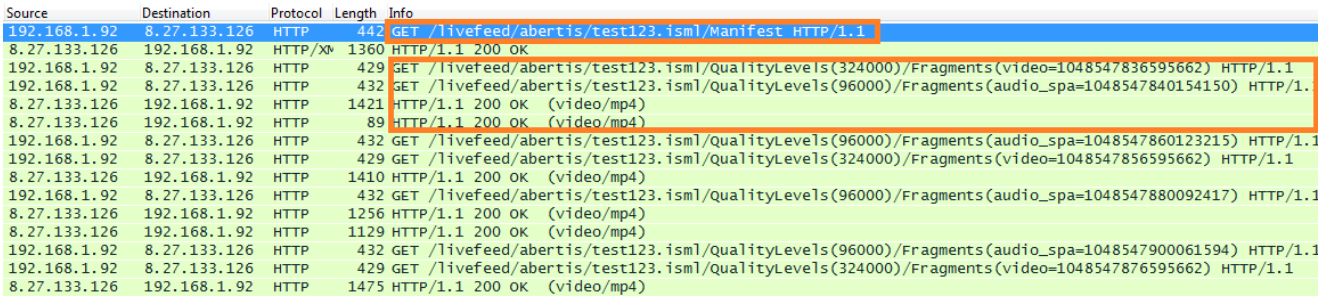


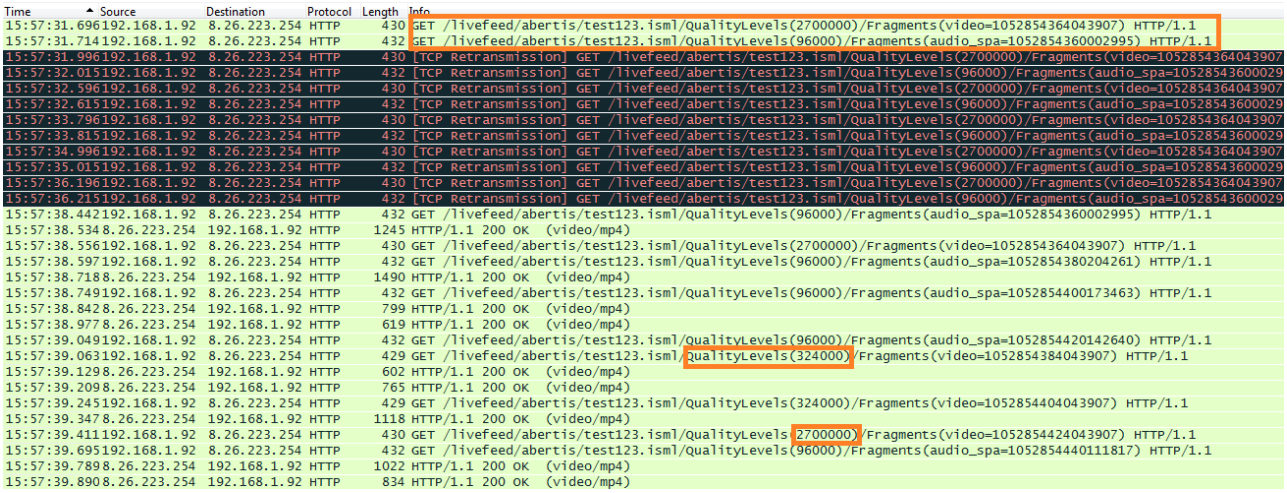
Figura 51. Captura de tráfico HSS

En la captura podemos ver la petición del Manifest al inicio de la reproducción y como se suceden las peticiones de los fragmentos de vídeo y audio por separado. Para el audio sólo hay una calidad configurada con un bit-rate de 96 kbps.

Para comenzar con estas pruebas realizamos un corte de larga duración con el objetivo de ver cuántos fragmentos se almacenan en el buffer, es decir, para analizar cuanto de cerca del directo funciona este reproductor.

Al realizar el corte se reproducen 8 segundos de vídeo antes de que el player deje de reproducir, lo que corresponde a 4 fragmentos (recordemos que la duración de un fragmento para esta propiedad es de 2 segundos).

Posteriormente, realizamos un corte de 6 segundos de duración, desde las 15:57:30 hasta las 15:57:36. Como es lógico no se corta la señal de vídeo (ya que se tienen fragmentos suficientes en el buffer para seguir reproduciendo).

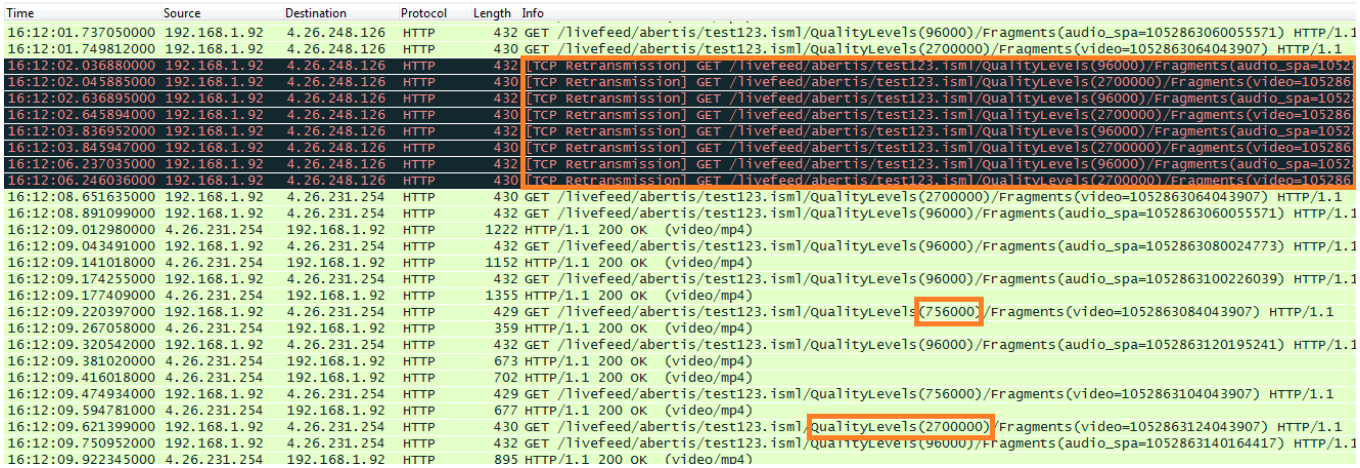


Time	Source	Destination	Protocol	Length	Info
15:57:31.696	192.168.1.92	8.26.223.254	HTTP	430	GET /livefeed/abertis/test123.isml/QualityLevels(2700000)/Fragments(video=1052854364043907) HTTP/1.1
15:57:31.714	192.168.1.92	8.26.223.254	HTTP	432	GET /livefeed/abertis/test123.isml/QualityLevels(96000)/Fragments(audio_spa=1052854360002995) HTTP/1.1
15:57:31.996	192.168.1.92	8.26.223.254	HTTP	430	[TCP Retransmission] GET /livefeed/abertis/test123.isml/QualityLevels(2700000)/Fragments(video=1052854364043907
15:57:32.015	192.168.1.92	8.26.223.254	HTTP	432	[TCP Retransmission] GET /livefeed/abertis/test123.isml/QualityLevels(96000)/Fragments(audio_spa=10528543600029
15:57:32.596	192.168.1.92	8.26.223.254	HTTP	430	[TCP Retransmission] GET /livefeed/abertis/test123.isml/QualityLevels(2700000)/Fragments(video=1052854364043907
15:57:32.615	192.168.1.92	8.26.223.254	HTTP	432	[TCP Retransmission] GET /livefeed/abertis/test123.isml/QualityLevels(96000)/Fragments(audio_spa=10528543600029
15:57:33.796	192.168.1.92	8.26.223.254	HTTP	430	[TCP Retransmission] GET /livefeed/abertis/test123.isml/QualityLevels(2700000)/Fragments(video=1052854364043907
15:57:33.815	192.168.1.92	8.26.223.254	HTTP	432	[TCP Retransmission] GET /livefeed/abertis/test123.isml/QualityLevels(96000)/Fragments(audio_spa=10528543600029
15:57:34.996	192.168.1.92	8.26.223.254	HTTP	430	[TCP Retransmission] GET /livefeed/abertis/test123.isml/QualityLevels(2700000)/Fragments(video=1052854364043907
15:57:35.015	192.168.1.92	8.26.223.254	HTTP	432	[TCP Retransmission] GET /livefeed/abertis/test123.isml/QualityLevels(96000)/Fragments(audio_spa=10528543600029
15:57:36.196	192.168.1.92	8.26.223.254	HTTP	430	[TCP Retransmission] GET /livefeed/abertis/test123.isml/QualityLevels(2700000)/Fragments(video=1052854364043907
15:57:36.215	192.168.1.92	8.26.223.254	HTTP	432	[TCP Retransmission] GET /livefeed/abertis/test123.isml/QualityLevels(96000)/Fragments(audio_spa=10528543600029
15:57:38.442	192.168.1.92	8.26.223.254	HTTP	432	GET /livefeed/abertis/test123.isml/QualityLevels(96000)/Fragments(audio_spa=1052854360002995) HTTP/1.1
15:57:38.534	8.26.223.254	192.168.1.92	HTTP	1245	HTTP/1.1 200 OK (video/mp4)
15:57:38.556	192.168.1.92	8.26.223.254	HTTP	430	GET /livefeed/abertis/test123.isml/QualityLevels(2700000)/Fragments(video=1052854364043907) HTTP/1.1
15:57:38.597	192.168.1.92	8.26.223.254	HTTP	432	GET /livefeed/abertis/test123.isml/QualityLevels(96000)/Fragments(audio_spa=1052854380204261) HTTP/1.1
15:57:38.718	8.26.223.254	192.168.1.92	HTTP	1490	HTTP/1.1 200 OK (video/mp4)
15:57:38.749	192.168.1.92	8.26.223.254	HTTP	432	GET /livefeed/abertis/test123.isml/QualityLevels(96000)/Fragments(audio_spa=1052854400173463) HTTP/1.1
15:57:38.842	8.26.223.254	192.168.1.92	HTTP	799	HTTP/1.1 200 OK (video/mp4)
15:57:38.977	8.26.223.254	192.168.1.92	HTTP	619	HTTP/1.1 200 OK (video/mp4)
15:57:39.049	192.168.1.92	8.26.223.254	HTTP	432	GET /livefeed/abertis/test123.isml/QualityLevels(96000)/Fragments(audio_spa=1052854420142640) HTTP/1.1
15:57:39.063	192.168.1.92	8.26.223.254	HTTP	429	GET /livefeed/abertis/test123.isml/QualityLevels(324000)/Fragments(video=1052854384043907) HTTP/1.1
15:57:39.129	8.26.223.254	192.168.1.92	HTTP	602	HTTP/1.1 200 OK (video/mp4)
15:57:39.209	8.26.223.254	192.168.1.92	HTTP	765	HTTP/1.1 200 OK (video/mp4)
15:57:39.245	192.168.1.92	8.26.223.254	HTTP	429	GET /livefeed/abertis/test123.isml/QualityLevels(324000)/Fragments(video=1052854404043907) HTTP/1.1
15:57:39.347	8.26.223.254	192.168.1.92	HTTP	1118	HTTP/1.1 200 OK (video/mp4)
15:57:39.411	192.168.1.92	8.26.223.254	HTTP	430	GET /livefeed/abertis/test123.isml/QualityLevels(2700000)/Fragments(video=1052854424043907) HTTP/1.1
15:57:39.695	192.168.1.92	8.26.223.254	HTTP	432	GET /livefeed/abertis/test123.isml/QualityLevels(96000)/Fragments(audio_spa=1052854440111817) HTTP/1.1
15:57:39.789	8.26.223.254	192.168.1.92	HTTP	1022	HTTP/1.1 200 OK (video/mp4)
15:57:39.890	8.26.223.254	192.168.1.92	HTTP	834	HTTP/1.1 200 OK (video/mp4)

Figura 52. Captura 1 HSS con el reproductor de Silverlight

Durante el periodo de corte se solicitan los fragmentos de audio y vídeo repetidamente. Al recuperar la conexión se retoma el tráfico habitual, sólo cabe destacar algún cambio en la calidad como se aprecia en la figura.

Al provocar un corte de 8 segundos, de las 16:12:00 a las 16:12:08, hay un pequeño corte (inferior a 1 segundo) y la reproducción continúa sin problemas.



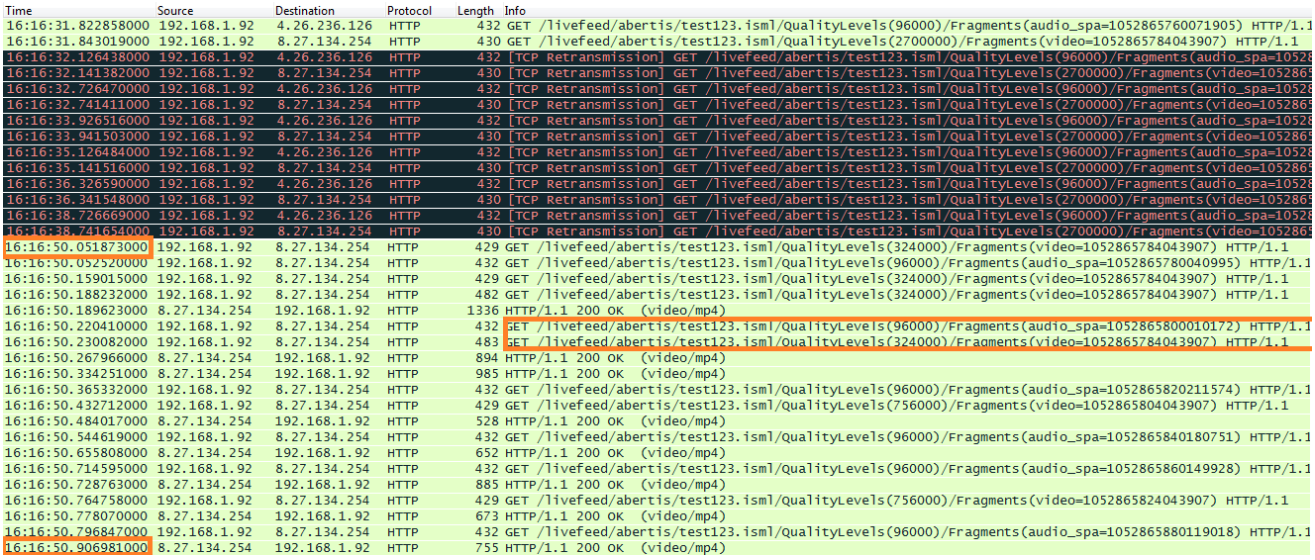
Time	Source	Destination	Protocol	Length	Info
16:12:01.737050000	192.168.1.92	4.26.248.126	HTTP	432	GET /livefeed/abertis/test123.isml/QualityLevels(96000)/Fragments(audio_spa=1052863060055571) HTTP/1.1
16:12:01.749812000	192.168.1.92	4.26.248.126	HTTP	430	GET /livefeed/abertis/test123.isml/QualityLevels(2700000)/Fragments(video=1052863064043907) HTTP/1.1
16:12:02.036880000	192.168.1.92	4.26.248.126	HTTP	432	[TCP Retransmission] GET /livefeed/abertis/test123.isml/QualityLevels(96000)/Fragments(audio_spa=1052
16:12:02.045885000	192.168.1.92	4.26.248.126	HTTP	430	[TCP Retransmission] GET /livefeed/abertis/test123.isml/QualityLevels(2700000)/Fragments(video=105286
16:12:02.636895000	192.168.1.92	4.26.248.126	HTTP	432	[TCP Retransmission] GET /livefeed/abertis/test123.isml/QualityLevels(96000)/Fragments(audio_spa=1052
16:12:02.645894000	192.168.1.92	4.26.248.126	HTTP	430	[TCP Retransmission] GET /livefeed/abertis/test123.isml/QualityLevels(2700000)/Fragments(video=105286
16:12:03.836952000	192.168.1.92	4.26.248.126	HTTP	432	[TCP Retransmission] GET /livefeed/abertis/test123.isml/QualityLevels(96000)/Fragments(audio_spa=1052
16:12:03.845947000	192.168.1.92	4.26.248.126	HTTP	430	[TCP Retransmission] GET /livefeed/abertis/test123.isml/QualityLevels(2700000)/Fragments(video=105286
16:12:06.237035000	192.168.1.92	4.26.248.126	HTTP	432	[TCP Retransmission] GET /livefeed/abertis/test123.isml/QualityLevels(96000)/Fragments(audio_spa=1052
16:12:06.246036000	192.168.1.92	4.26.248.126	HTTP	430	[TCP Retransmission] GET /livefeed/abertis/test123.isml/QualityLevels(2700000)/Fragments(video=105286
16:12:08.651635000	192.168.1.92	4.26.231.254	HTTP	430	GET /livefeed/abertis/test123.isml/QualityLevels(2700000)/Fragments(video=1052863064043907) HTTP/1.1
16:12:08.891099000	192.168.1.92	4.26.231.254	HTTP	432	GET /livefeed/abertis/test123.isml/QualityLevels(96000)/Fragments(audio_spa=1052863060055571) HTTP/1.1
16:12:09.012980000	4.26.231.254	192.168.1.92	HTTP	1222	HTTP/1.1 200 OK (video/mp4)
16:12:09.043491000	192.168.1.92	4.26.231.254	HTTP	432	GET /livefeed/abertis/test123.isml/QualityLevels(96000)/Fragments(audio_spa=1052863080024773) HTTP/1.1
16:12:09.141018000	4.26.231.254	192.168.1.92	HTTP	1152	HTTP/1.1 200 OK (video/mp4)
16:12:09.174255000	192.168.1.92	4.26.231.254	HTTP	432	GET /livefeed/abertis/test123.isml/QualityLevels(96000)/Fragments(audio_spa=1052863100226039) HTTP/1.1
16:12:09.177409000	4.26.231.254	192.168.1.92	HTTP	1355	HTTP/1.1 200 OK (video/mp4)
16:12:09.220397000	192.168.1.92	4.26.231.254	HTTP	429	GET /livefeed/abertis/test123.isml/QualityLevels(756000)/Fragments(video=1052863084043907) HTTP/1.1
16:12:09.267058000	4.26.231.254	192.168.1.92	HTTP	359	HTTP/1.1 200 OK (video/mp4)
16:12:09.320542000	192.168.1.92	4.26.231.254	HTTP	432	GET /livefeed/abertis/test123.isml/QualityLevels(96000)/Fragments(audio_spa=1052863120195241) HTTP/1.1
16:12:09.381020000	4.26.231.254	192.168.1.92	HTTP	673	HTTP/1.1 200 OK (video/mp4)
16:12:09.416018000	4.26.231.254	192.168.1.92	HTTP	702	HTTP/1.1 200 OK (video/mp4)
16:12:09.474934000	192.168.1.92	4.26.231.254	HTTP	429	GET /livefeed/abertis/test123.isml/QualityLevels(756000)/Fragments(video=1052863104043907) HTTP/1.1
16:12:09.594781000	4.26.231.254	192.168.1.92	HTTP	677	HTTP/1.1 200 OK (video/mp4)
16:12:09.621399000	192.168.1.92	4.26.231.254	HTTP	430	GET /livefeed/abertis/test123.isml/QualityLevels(2700000)/Fragments(video=1052863124043907) HTTP/1.1
16:12:09.750952000	192.168.1.92	4.26.231.254	HTTP	432	GET /livefeed/abertis/test123.isml/QualityLevels(96000)/Fragments(audio_spa=1052863140164417) HTTP/1.1
16:12:09.922345000	4.26.231.254	192.168.1.92	HTTP	895	HTTP/1.1 200 OK (video/mp4)

Figura 53. Captura 2 HSS con el reproductor de Silverlight

La captura es similar a la anterior. El comportamiento únicamente cambia en que al ser el corte más largo, el reproductor se queda sin fragmentos y hay un pequeño corte. De nuevo se puede apreciar un salto a una calidad inferior, en este caso, a una con un bit-rate de 756 kbps.

Realizamos más pruebas. Con un corte de 12 segundos, tenemos una situación parecida a la anterior pero con un corte de la señal de vídeo algo mayor. Al provocar un corte de 16 segundos, se pierde la señal de vídeo durante unos 10 segundos antes de recuperar la reproducción. En la captura se puede apreciar como en muy poco tiempo, el player hace muchas peticiones de fragmentos de audio y vídeo. Además, hay un cambio de calidad

en el vídeo recuperado que se prolonga más tiempo hasta recuperar de nuevo la calidad más alta.



Time	Source	Destination	Protocol	Length	Info
16:16:31.822858000	192.168.1.92	4.26.236.126	HTTP	432	GET /livefeed/abertis/test123.isml/QualityLevels(96000)/Fragments(audio_spa=1052865760071905) HTTP/1.1
16:16:31.843019000	192.168.1.92	8.27.134.254	HTTP	430	GET /livefeed/abertis/test123.isml/QualityLevels(2700000)/Fragments(video=1052865784043907) HTTP/1.1
16:16:32.126438000	192.168.1.92	4.26.236.126	HTTP	432	[TCP Retransmission] GET /livefeed/abertis/test123.isml/QualityLevels(96000)/Fragments(audio_spa=1052
16:16:32.141382000	192.168.1.92	8.27.134.254	HTTP	430	[TCP Retransmission] GET /livefeed/abertis/test123.isml/QualityLevels(2700000)/Fragments(video=105286
16:16:32.726470000	192.168.1.92	4.26.236.126	HTTP	432	[TCP Retransmission] GET /livefeed/abertis/test123.isml/QualityLevels(96000)/Fragments(audio_spa=1052
16:16:32.741411000	192.168.1.92	8.27.134.254	HTTP	430	[TCP Retransmission] GET /livefeed/abertis/test123.isml/QualityLevels(2700000)/Fragments(video=105286
16:16:33.926516000	192.168.1.92	4.26.236.126	HTTP	432	[TCP Retransmission] GET /livefeed/abertis/test123.isml/QualityLevels(96000)/Fragments(audio_spa=1052
16:16:33.941503000	192.168.1.92	8.27.134.254	HTTP	430	[TCP Retransmission] GET /livefeed/abertis/test123.isml/QualityLevels(2700000)/Fragments(video=105286
16:16:35.126484000	192.168.1.92	4.26.236.126	HTTP	432	[TCP Retransmission] GET /livefeed/abertis/test123.isml/QualityLevels(96000)/Fragments(audio_spa=1052
16:16:35.141516000	192.168.1.92	8.27.134.254	HTTP	430	[TCP Retransmission] GET /livefeed/abertis/test123.isml/QualityLevels(2700000)/Fragments(video=105286
16:16:36.326590000	192.168.1.92	4.26.236.126	HTTP	432	[TCP Retransmission] GET /livefeed/abertis/test123.isml/QualityLevels(96000)/Fragments(audio_spa=1052
16:16:36.341548000	192.168.1.92	8.27.134.254	HTTP	430	[TCP Retransmission] GET /livefeed/abertis/test123.isml/QualityLevels(2700000)/Fragments(video=105286
16:16:38.726669000	192.168.1.92	4.26.236.126	HTTP	432	[TCP Retransmission] GET /livefeed/abertis/test123.isml/QualityLevels(96000)/Fragments(audio_spa=1052
16:16:38.741654000	192.168.1.92	8.27.134.254	HTTP	430	[TCP Retransmission] GET /livefeed/abertis/test123.isml/QualityLevels(2700000)/Fragments(video=105286
16:16:50.051873000	192.168.1.92	8.27.134.254	HTTP	429	GET /livefeed/abertis/test123.isml/QualityLevels(324000)/Fragments(video=1052865784043907) HTTP/1.1
16:16:50.052520000	192.168.1.92	8.27.134.254	HTTP	432	GET /livefeed/abertis/test123.isml/QualityLevels(96000)/Fragments(audio_spa=1052865780040995) HTTP/1.1
16:16:50.159015000	192.168.1.92	8.27.134.254	HTTP	429	GET /livefeed/abertis/test123.isml/QualityLevels(324000)/Fragments(video=1052865784043907) HTTP/1.1
16:16:50.188232000	192.168.1.92	8.27.134.254	HTTP	482	GET /livefeed/abertis/test123.isml/QualityLevels(324000)/Fragments(video=1052865784043907) HTTP/1.1
16:16:50.189623000	8.27.134.254	192.168.1.92	HTTP	1336	HTTP/1.1 200 OK (video/mp4)
16:16:50.220410000	192.168.1.92	8.27.134.254	HTTP	432	GET /livefeed/abertis/test123.isml/QualityLevels(96000)/Fragments(audio_spa=1052865800010172) HTTP/1.1
16:16:50.230082000	192.168.1.92	8.27.134.254	HTTP	483	GET /livefeed/abertis/test123.isml/QualityLevels(324000)/Fragments(video=1052865784043907) HTTP/1.1
16:16:50.267966000	8.27.134.254	192.168.1.92	HTTP	894	HTTP/1.1 200 OK (video/mp4)
16:16:50.334251000	8.27.134.254	192.168.1.92	HTTP	985	HTTP/1.1 200 OK (video/mp4)
16:16:50.365332000	192.168.1.92	8.27.134.254	HTTP	432	GET /livefeed/abertis/test123.isml/QualityLevels(96000)/Fragments(audio_spa=1052865820211574) HTTP/1.1
16:16:50.432712000	192.168.1.92	8.27.134.254	HTTP	429	GET /livefeed/abertis/test123.isml/QualityLevels(756000)/Fragments(video=1052865804043907) HTTP/1.1
16:16:50.484017000	8.27.134.254	192.168.1.92	HTTP	528	HTTP/1.1 200 OK (video/mp4)
16:16:50.544619000	192.168.1.92	8.27.134.254	HTTP	432	GET /livefeed/abertis/test123.isml/QualityLevels(96000)/Fragments(audio_spa=1052865840180751) HTTP/1.1
16:16:50.655808000	8.27.134.254	192.168.1.92	HTTP	652	HTTP/1.1 200 OK (video/mp4)
16:16:50.714595000	192.168.1.92	8.27.134.254	HTTP	432	GET /livefeed/abertis/test123.isml/QualityLevels(96000)/Fragments(audio_spa=1052865860149928) HTTP/1.1
16:16:50.728763000	8.27.134.254	192.168.1.92	HTTP	885	HTTP/1.1 200 OK (video/mp4)
16:16:50.764758000	192.168.1.92	8.27.134.254	HTTP	429	GET /livefeed/abertis/test123.isml/QualityLevels(756000)/Fragments(video=1052865824043907) HTTP/1.1
16:16:50.778070000	8.27.134.254	192.168.1.92	HTTP	673	HTTP/1.1 200 OK (video/mp4)
16:16:50.796847000	192.168.1.92	8.27.134.254	HTTP	432	GET /livefeed/abertis/test123.isml/QualityLevels(96000)/Fragments(audio_spa=1052865880119018) HTTP/1.1
16:16:50.906981000	8.27.134.254	192.168.1.92	HTTP	755	HTTP/1.1 200 OK (video/mp4)

Figura 54. Captura 3 HSS con el reproductor de Silverlight

Finalmente, realizamos pruebas con cortes de 20, 22 y 24 segundos. Con un corte de 20 segundos ocurre el mismo comportamiento que en el caso mencionado anteriormente; pero al introducir un corte de 22 segundos o superior, el player se queda haciendo buffering y finalmente se cierra (estado "closed").

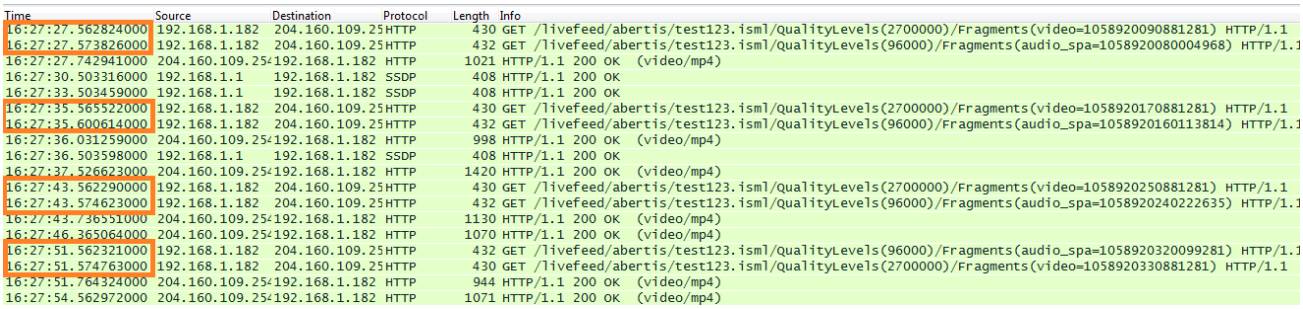
3.3.2.3. Pruebas HSS cambiando la duración del fragmento

Vamos a proceder a cambiar uno de los parámetros de configuración de Smooth Streaming, en este caso, la duración de un fragmento. Para ello vamos a emplear la misma propiedad de pruebas que en el caso anterior con el reproductor de Silverlight. Como comentamos anteriormente, para el caso de Ono no podemos cambiar la duración del fragmento, por lo que decidimos emplear únicamente el reproductor estándar para estas pruebas.

Para cambiar la duración de los chunks tenemos que acceder al codificador y en "*Settings*" nos permite establecer la duración de los fragmentos de audio y vídeo.

En las pruebas anteriores los fragmentos tenían una duración de 2 segundos (establecida por el estándar). En este caso procedemos a modificar la duración y la establecemos en 8 segundos.

Si analizamos tráfico normal veremos que se solicitan los fragmentos de audio y vídeo cada 8 segundos (duración fragmento), en vez de cada 2 segundos como en el caso anterior.



Time	Source	Destination	Protocol	Length	Info
16:27:27.562824000	192.168.1.182	204.160.109.25	HTTP	430	GET /livefeed/abertis/test123.isml/QualityLevels(2700000)/Fragments(video=1058920090881281) HTTP/1.1
16:27:27.573826000	192.168.1.182	204.160.109.25	HTTP	432	GET /livefeed/abertis/test123.isml/QualityLevels(96000)/Fragments(audio_spa=1058920080004968) HTTP/1.1
16:27:27.742941000	204.160.109.25	192.168.1.182	HTTP	1021	HTTP/1.1 200 OK (video/mp4)
16:27:30.503316000	192.168.1.1	192.168.1.182	SSDP	408	HTTP/1.1 200 OK
16:27:33.503459000	192.168.1.1	192.168.1.182	SSDP	408	HTTP/1.1 200 OK
16:27:35.565522000	192.168.1.182	204.160.109.25	HTTP	430	GET /livefeed/abertis/test123.isml/QualityLevels(2700000)/Fragments(video=1058920170881281) HTTP/1.1
16:27:35.600614000	192.168.1.182	204.160.109.25	HTTP	432	GET /livefeed/abertis/test123.isml/QualityLevels(96000)/Fragments(audio_spa=1058920160113814) HTTP/1.1
16:27:36.031259000	204.160.109.25	192.168.1.182	HTTP	998	HTTP/1.1 200 OK (video/mp4)
16:27:36.503598000	192.168.1.1	192.168.1.182	SSDP	408	HTTP/1.1 200 OK
16:27:37.526623000	204.160.109.25	192.168.1.182	HTTP	1420	HTTP/1.1 200 OK (video/mp4)
16:27:43.562290000	192.168.1.182	204.160.109.25	HTTP	430	GET /livefeed/abertis/test123.isml/QualityLevels(2700000)/Fragments(video=1058920250881281) HTTP/1.1
16:27:43.574623000	192.168.1.182	204.160.109.25	HTTP	432	GET /livefeed/abertis/test123.isml/QualityLevels(96000)/Fragments(audio_spa=1058920240222635) HTTP/1.1
16:27:43.736551000	204.160.109.25	192.168.1.182	HTTP	1130	HTTP/1.1 200 OK (video/mp4)
16:27:46.365064000	204.160.109.25	192.168.1.182	HTTP	1070	HTTP/1.1 200 OK (video/mp4)
16:27:51.562321000	192.168.1.182	204.160.109.25	HTTP	432	GET /livefeed/abertis/test123.isml/QualityLevels(96000)/Fragments(audio_spa=1058920320099281) HTTP/1.1
16:27:51.574763000	192.168.1.182	204.160.109.25	HTTP	430	GET /livefeed/abertis/test123.isml/QualityLevels(2700000)/Fragments(video=1058920330881281) HTTP/1.1
16:27:51.764324000	204.160.109.25	192.168.1.182	HTTP	944	HTTP/1.1 200 OK (video/mp4)
16:27:54.562972000	204.160.109.25	192.168.1.182	HTTP	1071	HTTP/1.1 200 OK (video/mp4)

Figura 55. Captura 1 HSS con el reproductor de Silverlight cambiando duración de los fragmentos

Sin embargo, hemos podido comprobar que el aumento de la duración de los fragmentos afecta negativamente al reproductor de Silverlight puesto que a pesar de comenzar a transmitir el vídeo correctamente al cabo de unos minutos el vídeo se para y el estado del reproductor pasa a "closed".

En Smooth Streaming se definió una duración típica de los fragmentos entre 2-4 segundos y, a pesar de que en el codificador podemos aumentar la duración de los mismos hasta 10 segundos, vemos como el reproductor no está configurado para poder soportar esta característica.

SMOOTH STREAMING			
PLATAFORMA/PLAYER	**TAMAÑO DEL BUFFER**	**DURACIÓN CHUNKS**	**MAXIMO CORTE SOPORTADO**
Ono	6 segundos	2 segundos	38 segundos
Silverlight	8 segundos	2 segundos	22 segundos
Silverlight	---	8 segundos	---

Tabla 6. Resumen pruebas Smooth Streaming

Por tanto tras esta batería de pruebas, hemos podido ver que el tamaño del buffer del reproductor de Silverlight es ligeramente superior al del reproductor de Ono, aunque en éste último el reproductor no se cierra tan pronto como el de Silverlight ante cortes de larga duración (característica del player).

3.3.2.4. Pruebas HSS cambiando la duración del DVR

A continuación vamos a explicar cómo podemos cambiar la duración del DVR tomando como ejemplo la propiedad de Smooth Streaming. La propiedad que hemos estado utilizando tenía un DVR de 10 minutos. Procedimos a cambiar el DVR para configurarlo con una duración de 1 minuto. Hay que destacar que en Smooth Streaming el DVR siempre debe ser igual o superior a 1 minuto.

Para configurar el DVR a un valor determinado, hay que parar la propiedad. Para ello accedimos al codificador, dónde paramos la publicación de la propiedad. Posteriormente

hay que acceder al AOM (Adaptive Origin Manager), un administrador de todas las propiedades de vídeo, dónde podemos modificar varios parámetros, entre ellos el DVR.

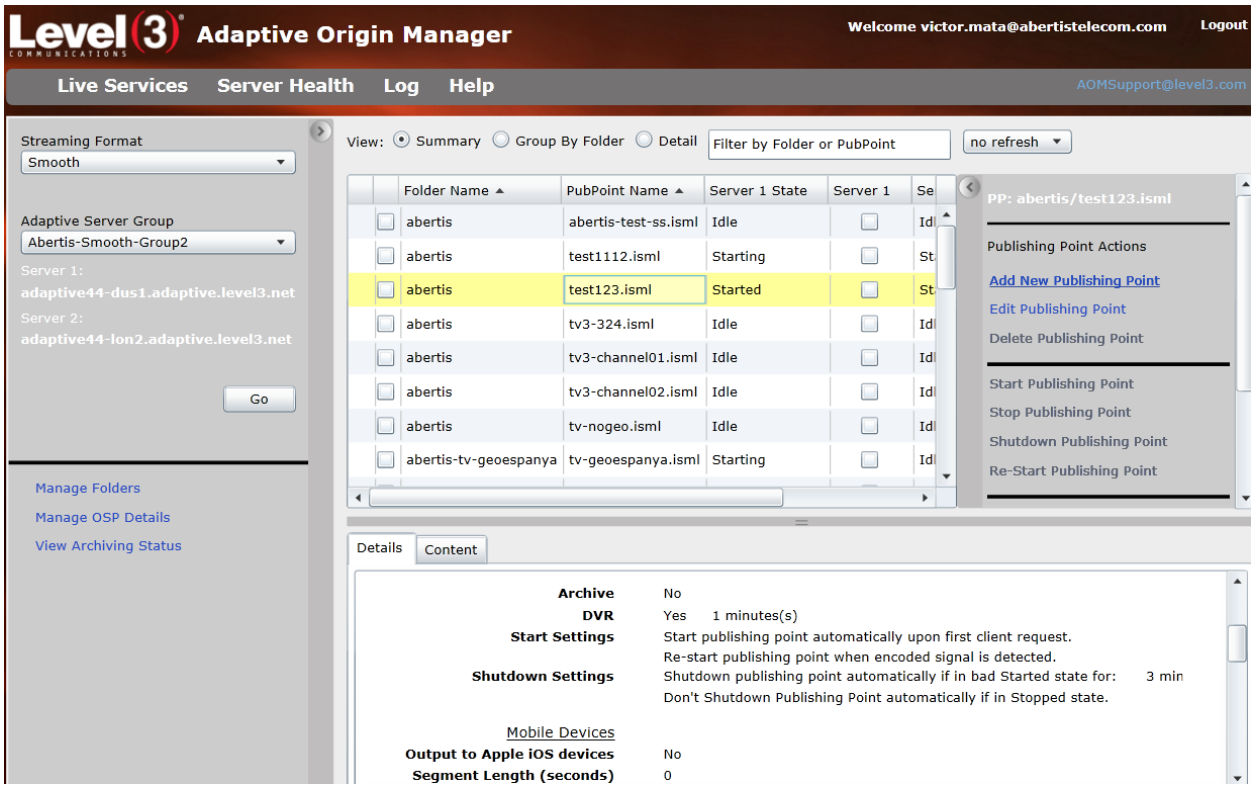


Figura 56. AOM (Adaptive Origin Manager)

Una vez que tenemos parada la propiedad, procedemos a editar el punto de publicación a través del panel que aparece en la parte derecha de la figura 56.

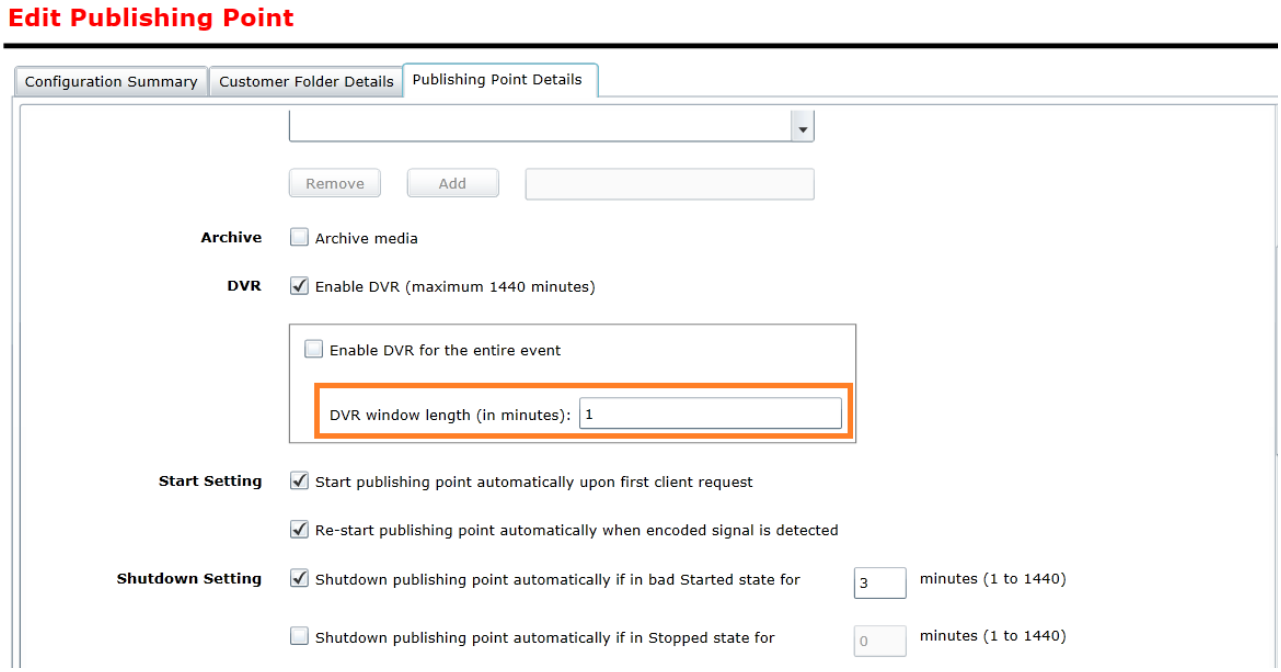


Figura 57. Cambio del parámetro DVR

El principal efecto que el usuario aprecia al bajar el DVR a 1 minuto, es que ahora sólo podemos dar marcha atrás (rebobinar) como máximo 1 minuto.

Se realizaron pruebas provocando cortes con el dispositivo de Albedo, obteniendo resultados muy similares o prácticamente iguales a los llevados a cabo sobre esta propiedad con el DVR anterior configurado con 10 minutos. Al provocar un corte de 8 segundos el vídeo se cortaba durante 2 segundos aproximadamente, para un corte de 16 segundos, el player se paraba durante unos 10 segundos; y para un corte más largo, superior a 22 segundos, la aplicación se cerraba dejando de reproducir definitivamente el vídeo.

Por tanto, en estos casos el cambio de DVR no influía en la reproducción del contenido, simplemente se aprecia el efecto en el tiempo que podemos retroceder el vídeo. Un caso idéntico ocurre para los protocolos HDS y MPEG-DASH por lo que decidimos no exponer de nuevo los casos en los posteriores apartados.

3.3.3. Pruebas protocolo HDS

Para realizar las pruebas del protocolo HDS (solución de Adobe) utilizamos un escenario idéntico al utilizado en HSS. Nos ayudamos del ordenador portátil y del dispositivo de Albedo para introducir los cortes.

Figura 58. Escenario HDS

En este caso vamos a utilizar un reproductor del proveedor de la CDN con el protocolo HDS. Además, como hemos hecho en casos anteriores, analizaremos el reproductor de Nubeox en PC que trabaja con tráfico HDS. Hay que destacar que no pudimos emplear el reproductor estándar de Adobe ya que había sido eliminado por dicha empresa de su portal para poderlo descargar. Por esta razón decidimos emplear un reproductor del proveedor de la CDN que nos garantiza resultados verosímiles.

3.3.3.1. Pruebas HDS con Nubeox

Para realizar las pruebas vamos a utilizar una propiedad de A3 que tiene establecidos la duración de un fragmento en 4 segundos y un DVR de 60 segundos.

En primer lugar capturamos el tráfico al iniciar la reproducción de un canal y observamos el siguiente comportamiento.

ServerDone...	Result	Protocol	Host	URL
15:40:51.313	200	HTTP	nubeoxlivegeo-live.hds.adaptive.level3.net	/hds/axnwhitehd/axnwhitehd.f4m
15:40:51.732	200	HTTP	nubeoxlivegeo-live.hds.adaptive.level3.net	/hds-live/axnwhitehd/_definst_/liveevent/livestream1.f4m
15:40:51.855	200	HTTP	nubeoxlivegeo-live.hds.adaptive.level3.net	/hds-live/axnwhitehd/_definst_/liveevent/livestream3.f4m
15:40:51.879	200	HTTP	nubeoxlivegeo-live.hds.adaptive.level3.net	/hds-live/axnwhitehd/_definst_/liveevent/livestream4.f4m
15:40:51.830	200	HTTP	nubeoxlivegeo-live.hds.adaptive.level3.net	/hds-live/axnwhitehd/_definst_/liveevent/livestream2.f4m
15:40:52.297	200	HTTP	nubeoxlivegeo-live.hds.adaptive.level3.net	/hds-live/streams/axnwhitehd/events/_definst_/liveevent/livestream4.drmmeta
15:40:52.276	200	HTTP	nubeoxlivegeo-live.hds.adaptive.level3.net	/hds-live/streams/axnwhitehd/events/_definst_/liveevent/livestream3.drmmeta
15:40:52.283	200	HTTP	nubeoxlivegeo-live.hds.adaptive.level3.net	/hds-live/streams/axnwhitehd/events/_definst_/liveevent/livestream1.drmmeta
15:40:52.296	200	HTTP	nubeoxlivegeo-live.hds.adaptive.level3.net	/hds-live/streams/axnwhitehd/events/_definst_/liveevent/livestream2.drmmeta
15:40:52.636	200	HTTP	nubeoxlivegeo-live.hds.adaptive.level3.net	/hds-live/streams/axnwhitehd/events/_definst_/liveevent/livestream1.bootstrap
15:40:53.063	200	HTTP	nubeoxlivegeo-live.hds.adaptive.level3.net	/hds-live/streams/axnwhitehd/events/_definst_/liveevent/livestream2.bootstrap
15:40:53.388	200	HTTP	nubeoxlivegeo-live.hds.adaptive.level3.net	/hds-live/streams/axnwhitehd/events/_definst_/liveevent/livestream3.bootstrap
15:40:53.689	200	HTTP	nubeoxlivegeo-live.hds.adaptive.level3.net	/hds-live/streams/axnwhitehd/events/_definst_/liveevent/livestream4.bootstrap

Figura 59. Captura 1 HDS con canal de la plataforma Nubeox

Podemos ver que en primer lugar se solicita el Manifest, y a partir de ahí se solicitan los sub-manifest de cada una de las cuatro calidades del stream. A continuación se descarga la información de seguridad (archivos DRM) y los archivos de bootstrap (equivalente al archivo .m3u8 de HLS). Estos últimos nos informan de la secuencia de los fragmentos y segmentos, lo que permite al player comenzar a solicitar los fragmentos de vídeo.

15:40:56.783	200	HTTP	nubeoxlivegeo-live.hds.adaptive.level3.net	/hds-live/streams/axnwhitehd/events/_definst_/liveevent/livestream1Seg12632-Frag1263191
	200	HTTP	Tunnel to	antena3.adobeaccess.vualto.com:443
15:40:57.123	200	HTTP	nubeoxlivegeo-live.hds.adaptive.level3.net	/hds-live/streams/axnwhitehd/events/_definst_/liveevent/livestream2.bootstrap
15:41:00.422	200	HTTP	nubeoxlivegeo-live.hds.adaptive.level3.net	/hds-live/streams/axnwhitehd/events/_definst_/liveevent/livestream2Seg12632-Frag1263192
15:40:58.854	200	HTTP	nqs.eu.nice264.com	/ping?time=50.01300000026822&pingTime=5&totalBytes=506722&dataType=0&code=4b7j9cha1z8j...
15:41:00.777	200	HTTP	nubeoxlivegeo-live.hds.adaptive.level3.net	/hds-live/streams/axnwhitehd/events/_definst_/liveevent/livestream3.bootstrap
15:41:00.988	200	HTTP	nqs.eu.nice264.com	/joinTime?eventTime=46.00300000049174&time=5999&code=4b7j9cha1z8js93b
15:41:05.045	200	HTTP	nubeoxlivegeo-live.hds.adaptive.level3.net	/hds-live/streams/axnwhitehd/events/_definst_/liveevent/livestream3Seg12632-Frag1263193
15:41:03.196	304	HTTP	nubeoxlivegeo-live.hds.adaptive.level3.net	/hds-live/streams/axnwhitehd/events/_definst_/liveevent/livestream3.bootstrap
15:41:04.087	200	HTTP	nqs.eu.nice264.com	/ping?time=50.9070000005885968pingTime=5&totalBytes=1186627&dataType=0&code=4b7j9cha1...
15:41:05.380	200	HTTP	nubeoxlivegeo-live.hds.adaptive.level3.net	/hds-live/streams/axnwhitehd/events/_definst_/liveevent/livestream4.bootstrap
15:41:09.545	200	HTTP	nubeoxlivegeo-live.hds.adaptive.level3.net	/hds-live/streams/axnwhitehd/events/_definst_/liveevent/livestream4Seg12632-Frag1263194
15:41:09.282	200	HTTP	nqs.eu.nice264.com	/ping?time=52.0230000000447&pingTime=5&totalBytes=2280009&dataType=0&code=4b7j9cha1z8j...

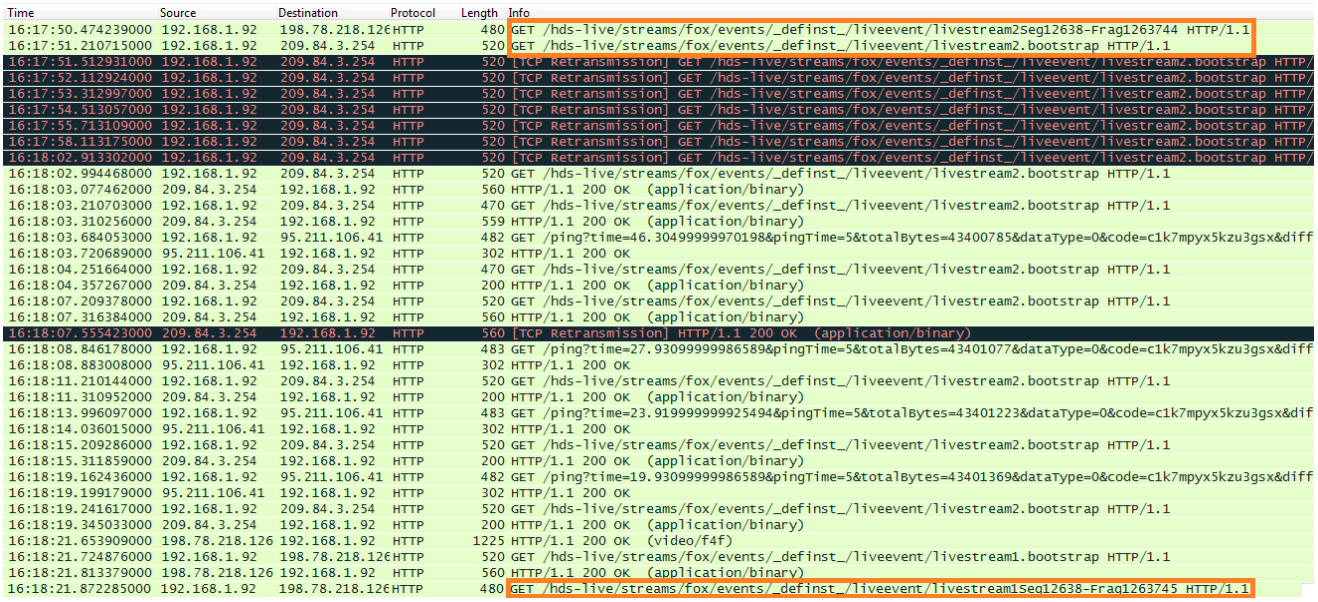
Figura 60. Captura 2 HDS con canal de la plataforma Nubeox

Podemos ver que todos los fragmentos solicitados corresponden al mismo segmento y que es el número del fragmento el que va aumentando de forma secuencial. Las calidades van variando por factores que el usuario no controla (puede ser por el estado de la red, por la carga de CPU,…). Al comienzo de la reproducción siempre se suele comenzar por la calidad más baja y si las condiciones lo permiten, se va subiendo hasta la mejor de las calidades.

Nubeox se apoya en un servidor que proporciona Adobe, denominado AMS (Adobe Media Server), que lleva instalado Apache que se encarga de empaquetar el contenido en HLS y HDS. A través de la consola que gestiona este servidor se puede comprobar que los fragmentos tienen una duración de 4 segundos, mientras que cada segmento tiene una duración de 400 segundos, es decir, analizando una captura de mayor duración podríamos comprobar que hay 100 fragmentos asociados a un único segmento.

Como en casos anteriores, realizamos un corte de larga duración para ver el tiempo que el reproductor es capaz de seguir proporcionando señal de vídeo y, por tanto, ver cuántos fragmentos hay almacenados en el buffer. Desde el momento de provocar el corte hasta que se para la reproducción transcurren unos 6 segundos (tiempo similar al de Smooth Streaming). Por tanto en el buffer se guardan en torno a 1/2 fragmentos.

Realizando un corte de duración 4 segundos, el contenido de vídeo no se corta. Si vamos aumentando el tiempo de corte, vemos como llega un momento en el que el player ya no tiene más fragmentos que reproducir y la señal se corta. Mostramos una captura de un corte de 8 segundos, desde las 16:17:50 a las 16:17:58.



Time	Source	Destination	Protocol	Length	Info
16:17:50.474239000	192.168.1.92	198.78.218.126	HTTP	480	GET /hds-live/streams/fox/events/_definst_/liveevent/livestream2Seg12638-Frag1263744 HTTP/1.1
16:17:51.210715000	192.168.1.92	209.84.3.254	HTTP	520	GET /hds-live/streams/fox/events/_definst_/liveevent/livestream2.bootstrap HTTP/1.1
16:17:51.512931000	192.168.1.92	209.84.3.254	HTTP	520	[TCP Retransmission] GET /hds-live/streams/fox/events/_definst_/liveevent/livestream2.bootstrap HTTP/
16:17:52.112924000	192.168.1.92	209.84.3.254	HTTP	520	[TCP Retransmission] GET /hds-live/streams/fox/events/_definst_/liveevent/livestream2.bootstrap HTTP/
16:17:53.312997000	192.168.1.92	209.84.3.254	HTTP	520	[TCP Retransmission] GET /hds-live/streams/fox/events/_definst_/liveevent/livestream2.bootstrap HTTP/
16:17:54.513057000	192.168.1.92	209.84.3.254	HTTP	520	[TCP Retransmission] GET /hds-live/streams/fox/events/_definst_/liveevent/livestream2.bootstrap HTTP/
16:17:55.713109000	192.168.1.92	209.84.3.254	HTTP	520	[TCP Retransmission] GET /hds-live/streams/fox/events/_definst_/liveevent/livestream2.bootstrap HTTP/
16:17:58.113175000	192.168.1.92	209.84.3.254	HTTP	520	[TCP Retransmission] GET /hds-live/streams/fox/events/_definst_/liveevent/livestream2.bootstrap HTTP/
16:18:02.913302000	192.168.1.92	209.84.3.254	HTTP	520	[TCP Retransmission] GET /hds-live/streams/fox/events/_definst_/liveevent/livestream2.bootstrap HTTP/
16:18:02.994468000	192.168.1.92	209.84.3.254	HTTP	520	GET /hds-live/streams/fox/events/_definst_/liveevent/livestream2.bootstrap HTTP/1.1
16:18:03.077462000	209.84.3.254	192.168.1.92	HTTP	560	HTTP/1.1 200 OK (application/binary)
16:18:03.210703000	192.168.1.92	209.84.3.254	HTTP	470	GET /hds-live/streams/fox/events/_definst_/liveevent/livestream2.bootstrap HTTP/1.1
16:18:03.310256000	209.84.3.254	192.168.1.92	HTTP	559	HTTP/1.1 200 OK (application/binary)
16:18:03.684053000	192.168.1.92	95.211.106.41	HTTP	482	GET /ping?time=46.30499999970198&pingTime=5&totalBytes=43400785&dataType=0&code=c1k7mpyx5kzu3gsx&diff
16:18:03.720689000	95.211.106.41	192.168.1.92	HTTP	302	HTTP/1.1 200 OK
16:18:04.251664000	192.168.1.92	209.84.3.254	HTTP	470	GET /hds-live/streams/fox/events/_definst_/liveevent/livestream2.bootstrap HTTP/1.1
16:18:04.357267000	209.84.3.254	192.168.1.92	HTTP	200	HTTP/1.1 200 OK (application/binary)
16:18:07.209378000	192.168.1.92	209.84.3.254	HTTP	520	GET /hds-live/streams/fox/events/_definst_/liveevent/livestream2.bootstrap HTTP/1.1
16:18:07.316384000	209.84.3.254	192.168.1.92	HTTP	560	HTTP/1.1 200 OK (application/binary)
16:18:07.555423000	209.84.3.254	192.168.1.92	HTTP	560	[TCP Retransmission] HTTP/1.1 200 OK (application/binary)
16:18:08.846178000	192.168.1.92	95.211.106.41	HTTP	483	GET /ping?time=27.93099999986589&pingTime=5&totalBytes=43401077&dataType=0&code=c1k7mpyx5kzu3gsx&diff
16:18:08.883008000	95.211.106.41	192.168.1.92	HTTP	302	HTTP/1.1 200 OK
16:18:11.210144000	192.168.1.92	209.84.3.254	HTTP	520	GET /hds-live/streams/fox/events/_definst_/liveevent/livestream2.bootstrap HTTP/1.1
16:18:11.310952000	209.84.3.254	192.168.1.92	HTTP	200	HTTP/1.1 200 OK (application/binary)
16:18:13.996097000	192.168.1.92	95.211.106.41	HTTP	483	GET /ping?time=23.9199999999925494&pingTime=5&totalBytes=43401223&dataType=0&code=c1k7mpyx5kzu3gsx&dif
16:18:14.036015000	95.211.106.41	192.168.1.92	HTTP	302	HTTP/1.1 200 OK
16:18:15.209286000	192.168.1.92	209.84.3.254	HTTP	520	GET /hds-live/streams/fox/events/_definst_/liveevent/livestream2.bootstrap HTTP/1.1
16:18:15.311859000	209.84.3.254	192.168.1.92	HTTP	200	HTTP/1.1 200 OK (application/binary)
16:18:19.162436000	192.168.1.92	95.211.106.41	HTTP	482	GET /ping?time=19.93099999986589&pingTime=5&totalBytes=43401369&dataType=0&code=c1k7mpyx5kzu3gsx&diff
16:18:19.199179000	95.211.106.41	192.168.1.92	HTTP	302	HTTP/1.1 200 OK
16:18:19.241617000	192.168.1.92	209.84.3.254	HTTP	520	GET /hds-live/streams/fox/events/_definst_/liveevent/livestream2.bootstrap HTTP/1.1
16:18:19.345033000	209.84.3.254	192.168.1.92	HTTP	200	HTTP/1.1 200 OK (application/binary)
16:18:21.653909000	198.78.218.126	192.168.1.92	HTTP	1225	HTTP/1.1 200 OK (video/f4f)
16:18:21.724876000	192.168.1.92	198.78.218.126	HTTP	520	GET /hds-live/streams/fox/events/_definst_/liveevent/livestream1.bootstrap HTTP/1.1
16:18:21.813379000	198.78.218.126	192.168.1.92	HTTP	560	HTTP/1.1 200 OK (application/binary)
16:18:21.872285000	192.168.1.92	198.78.218.126	HTTP	480	GET /hds-live/streams/fox/events/_definst_/liveevent/livestream1Seg12638-Frag1263745 HTTP/1.1

Figura 61. Captura 3 HDS con canal de la plataforma Nubeox

Podemos ver que el último fragmento solicitado antes de provocar el corte es el 1263744, durante el periodo del corte hay una serie de retransmisiones de la petición del archivo de bootstrap; el player por su parte reproduce los fragmentos de vídeo que almacena en el buffer hasta que no tiene más, que es cuando se produce el corte en la reproducción. Cuando se reanuda la conexión, se siguen solicitando fragmentos, comenzando por el siguiente, el 1263745, como se puede apreciar en la figura anterior.

El tiempo en el que el reproductor no presenta señal va aumentando en duración a medida que aumentamos la duración del corte provocado. Con un corte provocado de 20 segundos, el player se para durante unos 15 segundos aproximadamente. Mientras que para un corte de unos 50 segundos, el player está parado durante unos 40 segundos. Comprobamos realizando diversas pruebas que el player es capaz de continuar con la reproducción a pesar de presentar cortes tan duraderos.

3.3.3.2. Pruebas HDS con reproductor de la CDN

Para poder trabajar con el reproductor de HDS necesitábamos pedir una propiedad (ya que no disponíamos de ninguna de pruebas) al proveedor de servicios de CDN.

El proceso que seguimos para solicitar la propiedad HDS serviría para pedir cualquier otra propiedad de uno de los protocolos de transmisión de vídeo estudiados en el proyecto. Pedimos una propiedad para pruebas especificando el protocolo, los perfiles de codificación y el resto de parámetros relativos a HDS (DVR, duración del fragmento y el segmento). Para ello se rellenaba un formulario y se enviaba al proveedor de servicios de CDN.

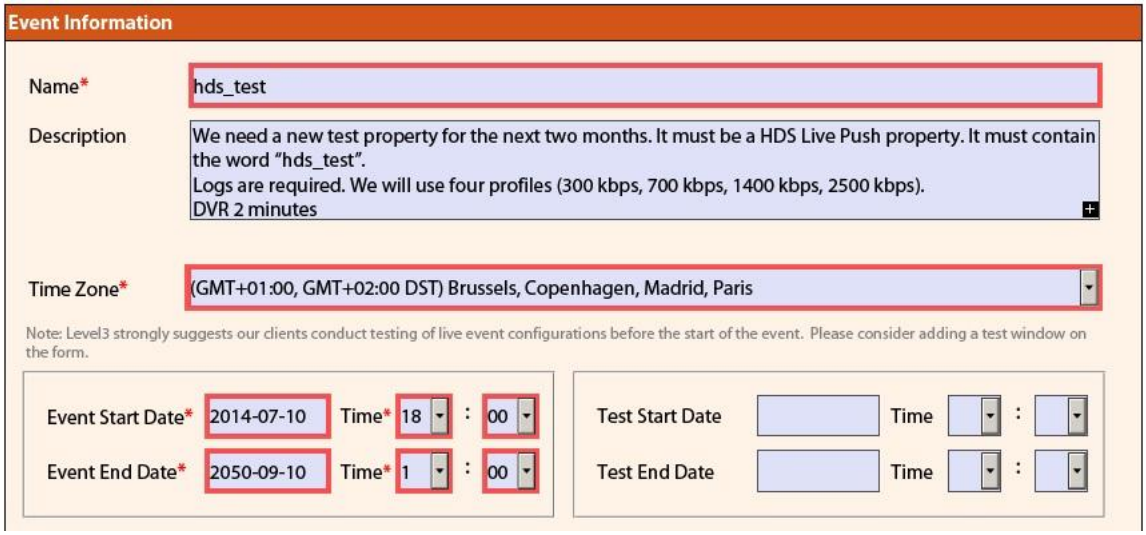
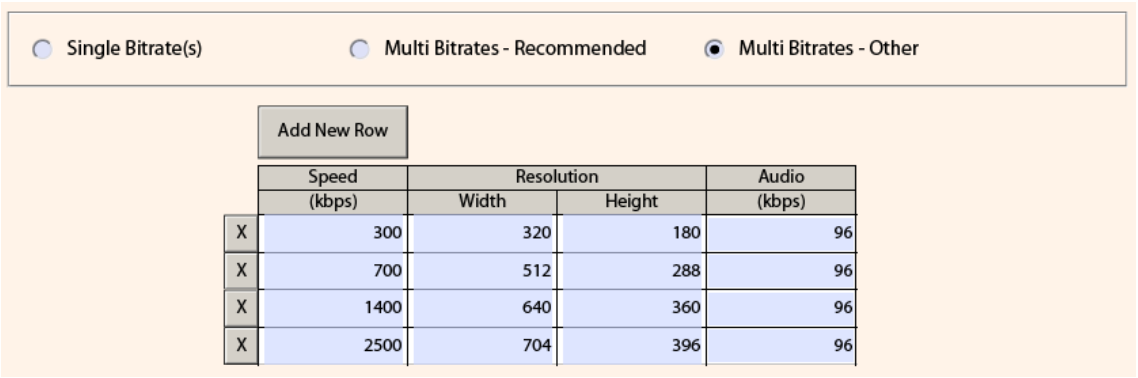


Figura 62. Parte del formulario enviado al proveedor de servicios de CDN

Al especificar las calidades se indicaba tanto el bit-rate como la resolución requerida.



	Speed (kbps)	Resolution Width	Resolution Height	Audio (kbps)
X	300	320	180	96
X	700	512	288	96
X	1400	640	360	96
X	2500	704	396	96

Figura 63. Especificación de calidades en el formulario

Una vez que el proveedor de servicios de CDN nos proporciona los puntos de publicación de la propiedad en su servidor, tenemos que configurarla en nuestro codificador y publicarla en el mismo. Para ello, definimos un punto de publicación y añadimos las 4 calidades correspondientes en el codificador. Es muy importante marcar en las opciones de trascodificación del equipo los bit-rates que queremos de cada una de las calidades (puesto que estas serán las que finalmente se codifiquen).

Dentro del AOS, el Adobe FMS (Flash Media Server) se encarga de juntar todas las calidades para cuando se vaya a reproducir el contenido se puedan producir los cambios de calidad de forma adaptativa.

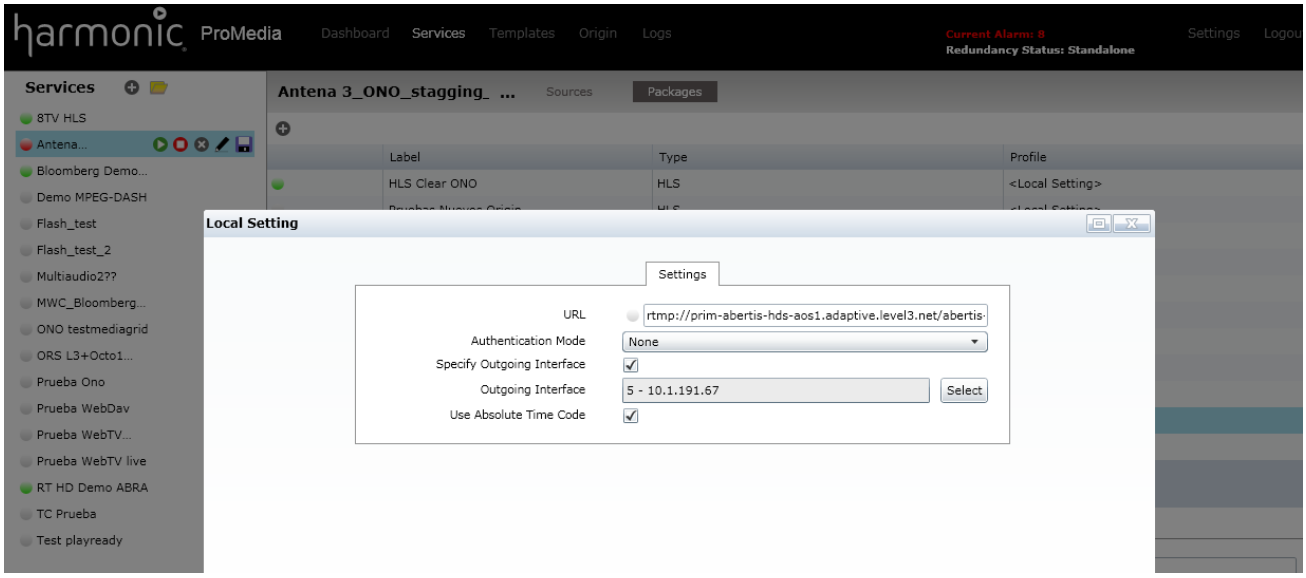


Figura 64. Codificador de Harmonic

Si accedemos al Manifest a través de la URL pública:

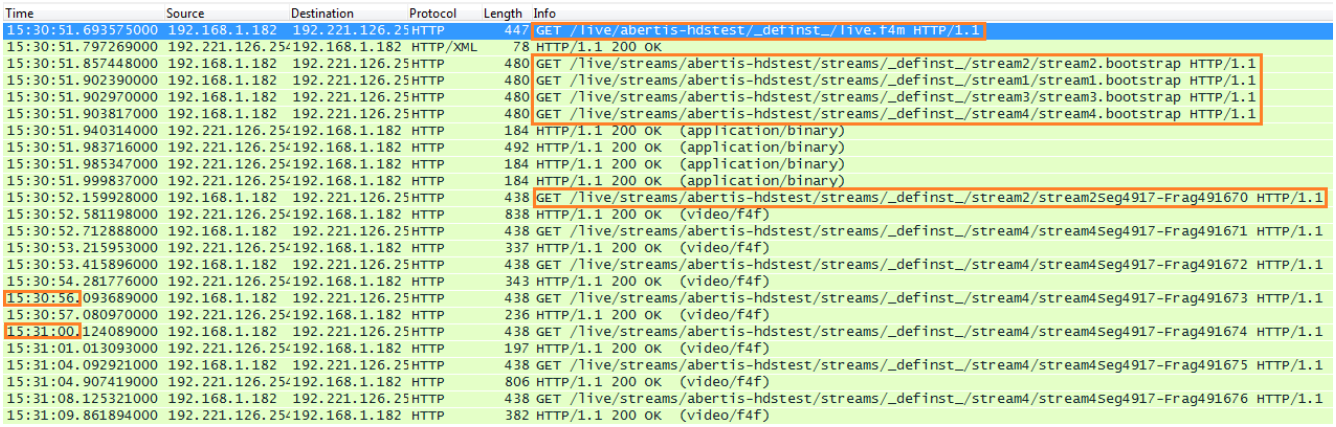
http://abertistest-live.hds.adaptive.level3.net/live/abertis-hdstest/_definst_/live.f4m


```
<manifest xmlns="http://ns.adobe.com/f4m/1.0">
  <id>abertis-hdstest/events/_definst_/live</id>
  <mimeType></mimeType>
  <streamType>live</streamType>
  <duration>0</duration>
  <dvrInfo endOffset="120"></dvrInfo>
  <media streamId="stream1" bitrate="700" url="../../streams/abertis-hdstest/streams/_definst_/stream1/stream1" bootstrapInfoId="bootstrap9578">
    <metadata>
      AgAKb25NZXRhRGF0YQgAAAAAAAhkdXJhdGlvbgBAGAQYk3S8agAFd21kdGgAQHQAAAAAAAAABmhlaWdodABAZoAAAAAAAAAAMdmlkZW9jb2RlY2lkAgAESDI2NAAMYXVkaW9jb2RlY2lkAgAEbXA0YQAKYXZjcHJvZ
    </metadata>
  </media>
  <media streamId="stream2" bitrate="300" url="../../streams/abertis-hdstest/streams/_definst_/stream2/stream2" bootstrapInfoId="bootstrap5862">
    <metadata>
      AgAKb25NZXRhRGF0YQgAAAAAAAhkdXJhdGlvbgBAGAQYk3S8agAFd21kdGgAQIAAAAAAAAAABmhlaWdodABAcgAAAAAAAAAAMdmlkZW9jb2RlY2lkAgAESDI2NAAMYXVkaW9jb2RlY2lkAgAEbXA0YQAKYXZjcHJvZ
    </metadata>
  </media>
  <media streamId="stream3" bitrate="1400" url="../../streams/abertis-hdstest/streams/_definst_/stream3/stream3" bootstrapInfoId="bootstrap5909">
    <metadata>
      AgAKb25NZXRhRGF0YQgAAAAAAAhkdXJhdGlvbgBAGAQYk3S8agAFd21kdGgAQIQAAAAAAAAABmhlaWdodABAdoAAAAAAAAAAMdmlkZW9jb2RlY2lkAgAESDI2NAAMYXVkaW9jb2RlY2lkAgAEbXA0YQAKYXZjcHJvZ
    </metadata>
  </media>
  <media streamId="stream4" bitrate="2500" url="../../streams/abertis-hdstest/streams/_definst_/stream4/stream4" bootstrapInfoId="bootstrap2876">
    <metadata>
      AgAKb25NZXRhRGF0YQgAAAAAAAhkdXJhdGlvbgBAGAQYk3S8agAFd21kdGgAQIYAAAAAAAAABmhlaWdodABAeMAAAAAAAAAAMdmlkZW9jb2RlY2lkAgAESDI2NAAMYXVkaW9jb2RlY2lkAgAEbXA0YQAKYXZjcHJvZ
    </metadata>
  </media>
  <bootstrapInfo profile="named" url="../../streams/abertis-hdstest/streams/_definst_/stream1/stream1.bootstrap" id="bootstrap9578"></bootstrapInfo>
  <bootstrapInfo profile="named" url="../../streams/abertis-hdstest/streams/_definst_/stream2/stream2.bootstrap" id="bootstrap5862"></bootstrapInfo>
  <bootstrapInfo profile="named" url="../../streams/abertis-hdstest/streams/_definst_/stream3/stream3.bootstrap" id="bootstrap5909"></bootstrapInfo>
  <bootstrapInfo profile="named" url="../../streams/abertis-hdstest/streams/_definst_/stream4/stream4.bootstrap" id="bootstrap2876"></bootstrapInfo>
</manifest>
```


Figura 65. Manifest HDS

En el manifest aparece la información del DVR, para esta propiedad es de 120 segundos, y a continuación cada una de las 4 calidades de la propiedad con su bit-rate correspondiente. Más abajo aparece la información de los segmentos de bootstrap, por cada calidad aparecen 1 archivo de inicialización diferente.

Si iniciamos la reproducción de nuestra propiedad y hacemos una captura podemos ver el comportamiento inicial.



Time	Source	Destination	Protocol	Length	Info
15:30:51.693575000	192.168.1.182	192.221.126.25	HTTP	447	GET /live/abertis-hdstest/_definst_/live.f4m HTTP/1.1
15:30:51.797269000	192.221.126.25	192.168.1.182	HTTP/XML	78	HTTP/1.1 200 OK
15:30:51.857448000	192.168.1.182	192.221.126.25	HTTP	480	GET /live/streams/abertis-hdstest/streams/_definst_/stream2/stream2.bootstrap HTTP/1.1
15:30:51.902390000	192.168.1.182	192.221.126.25	HTTP	480	GET /live/streams/abertis-hdstest/streams/_definst_/stream1/stream1.bootstrap HTTP/1.1
15:30:51.902970000	192.168.1.182	192.221.126.25	HTTP	480	GET /live/streams/abertis-hdstest/streams/_definst_/stream3/stream3.bootstrap HTTP/1.1
15:30:51.903817000	192.168.1.182	192.221.126.25	HTTP	480	GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4.bootstrap HTTP/1.1
15:30:51.940314000	192.221.126.25	192.168.1.182	HTTP	184	HTTP/1.1 200 OK (application/binary)
15:30:51.983716000	192.221.126.25	192.168.1.182	HTTP	492	HTTP/1.1 200 OK (application/binary)
15:30:51.985347000	192.221.126.25	192.168.1.182	HTTP	184	HTTP/1.1 200 OK (application/binary)
15:30:51.999837000	192.221.126.25	192.168.1.182	HTTP	184	HTTP/1.1 200 OK (application/binary)
15:30:52.159928000	192.168.1.182	192.221.126.25	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream2/stream2Seg4917-Frag491670 HTTP/1.1
15:30:52.581198000	192.221.126.25	192.168.1.182	HTTP	838	HTTP/1.1 200 OK (video/f4f)
15:30:52.712888000	192.168.1.182	192.221.126.25	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4917-Frag491671 HTTP/1.1
15:30:53.215953000	192.221.126.25	192.168.1.182	HTTP	337	HTTP/1.1 200 OK (video/f4f)
15:30:53.415896000	192.168.1.182	192.221.126.25	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4917-Frag491672 HTTP/1.1
15:30:54.281776000	192.221.126.25	192.168.1.182	HTTP	343	HTTP/1.1 200 OK (video/f4f)
15:30:56.093689000	192.168.1.182	192.221.126.25	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4917-Frag491673 HTTP/1.1
15:30:57.080970000	192.221.126.25	192.168.1.182	HTTP	236	HTTP/1.1 200 OK (video/f4f)
15:31:00.124089000	192.168.1.182	192.221.126.25	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4917-Frag491674 HTTP/1.1
15:31:01.013093000	192.221.126.25	192.168.1.182	HTTP	197	HTTP/1.1 200 OK (video/f4f)
15:31:04.092921000	192.168.1.182	192.221.126.25	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4917-Frag491675 HTTP/1.1
15:31:04.907419000	192.221.126.25	192.168.1.182	HTTP	806	HTTP/1.1 200 OK (video/f4f)
15:31:08.125321000	192.168.1.182	192.221.126.25	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4917-Frag491676 HTTP/1.1
15:31:09.861894000	192.221.126.25	192.168.1.182	HTTP	382	HTTP/1.1 200 OK (video/f4f)

Figura 66. Captura del inicio de la reproducción en canal HDS

Se comienza pidiendo el Manifest (archivo .f4m) a la CDN, una vez obtenida la respuesta se solicitan los 4 bootstrap correspondientes a las diferentes calidades y se comienza con el primer fragmento de vídeo. Normalmente cuando se comienza una reproducción, el primer fragmento pertenece a la calidad más baja (en este caso la correspondiente al stream 2, en el codificador está establecido que el stream 2 es la calidad con el bit-rate más bajo: 300 kbps) y ya el siguiente fragmento vemos que está referido a la calidad más alta (stream 4). Los primeros 3/4 fragmentos se solicitan en un corto intervalo de tiempo de apenas 4 segundos, pero a partir del cuarto fragmento el intervalo entre ellos es de 4 segundos (duración de un fragmento).

En el reproductor que vamos a utilizar se puede ver la evolución del buffer (medida en segundos). Se puede ver como al iniciar la reproducción el buffer se va llenando y va variando su capacidad según se va reproduciendo el contenido. En condiciones normales (es decir sin introducir ninguna imperfección en la señal) el buffer está variando entre los 7.7 y los 11 segundos aproximadamente.

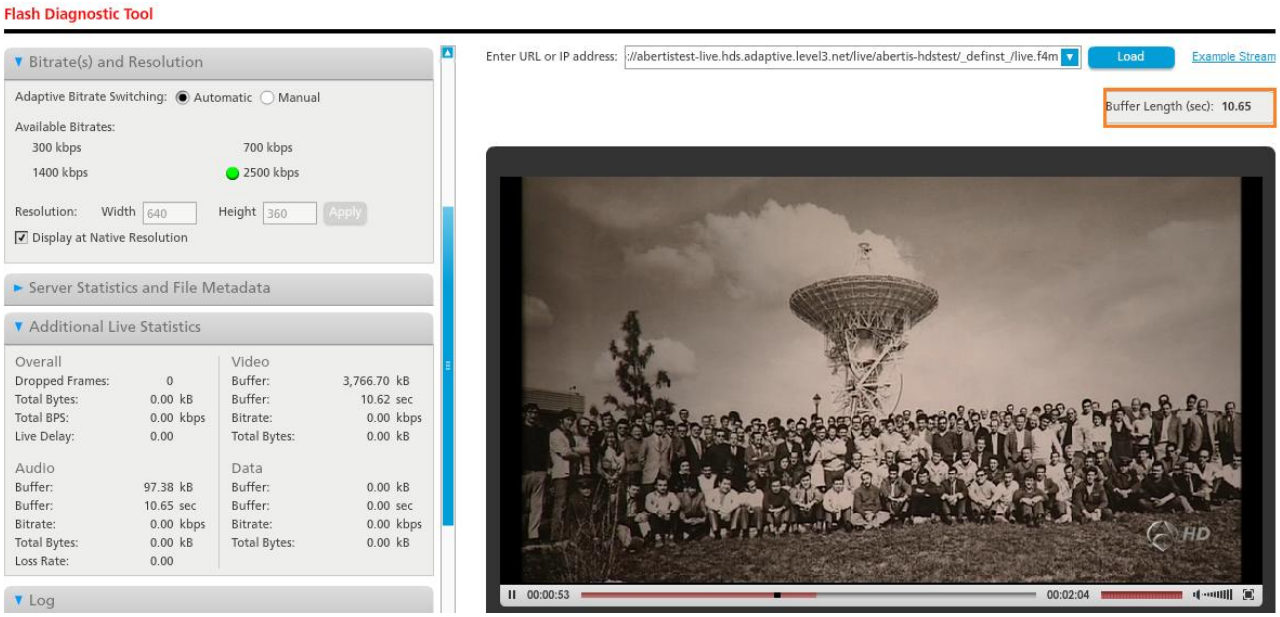


Figura 67. Reproductor para pruebas HDS

Como en todos los casos anteriores si provocamos un corte muy grande, el buffer comienza a disminuir y cuando ya no hay más fragmentos (segundos de vídeo) almacenados, la reproducción se corta. Aproximadamente el tamaño máximo que almacena el buffer es de dos/tres fragmentos HDS (lo que equivaldría a 8/12 segundos de vídeo).

A continuación siguiendo la misma metodología que en casos anteriores vamos provocando cortes con el dispositivo Net.Storm aumentando el tiempo de los mismos.

Al provocar un corte de 4 segundos, no se producen interrupciones en la reproducción, el buffer comienza a disminuir pero en cuanto se recupera la conexión se alcanzan de nuevo los valores habituales del buffer. Además, no hay ningún cambio de calidad.

Al introducir un corte de 8 segundos tampoco se para la reproducción, el buffer baja hasta 1 segundo aproximadamente y a partir de ese momento comienza a recuperarse.

Figura 68. Disminución del tamaño del buffer en HDS

En la imagen anterior podemos ver como el buffer ha bajado su tamaño habitual debido a uno de los cortes.

Al provocar un corte de 12 segundos, desde las 15:56:22 a las 15:56:34, ya observamos un corte en la reproducción de una duración aproximada de 3 segundos. Cuando se recupera el vídeo hay algún cambio de calidad hasta que se vuelve a estabilizar. Ese cambio viene ligado a un cambio en la resolución (en el tamaño de la ventana de vídeo). Durante las pruebas hemos podido apreciar el cambio en bastantes ocasiones.

Time	Source	Destination	Protocol	Length	Info
15:56:16.341387000	192.168.1.182	209.84.3.254	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4921-Frag492053 HTTP/1.1
15:56:17.047565000	209.84.3.254	192.168.1.182	HTTP	1356	HTTP/1.1 200 OK (video/f4f)
15:56:20.340629000	192.168.1.182	209.84.3.254	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4921-Frag492054 HTTP/1.1
15:56:21.223151000	209.84.3.254	192.168.1.182	HTTP	201	HTTP/1.1 200 OK (video/f4f)
15:56:24.341264000	192.168.1.182	209.84.3.254	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4921-Frag492055 HTTP/1.1
15:56:24.640640000	192.168.1.182	209.84.3.254	HTTP	438	[TCP Retransmission] GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4921-Frag4
15:56:25.240654000	192.168.1.182	209.84.3.254	HTTP	438	[TCP Retransmission] GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4921-Frag4
15:56:26.440670000	192.168.1.182	209.84.3.254	HTTP	438	[TCP Retransmission] GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4921-Frag4
15:56:27.640743000	192.168.1.182	209.84.3.254	HTTP	438	[TCP Retransmission] GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4921-Frag4
15:56:28.840746000	192.168.1.182	209.84.3.254	HTTP	438	[TCP Retransmission] GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4921-Frag4
15:56:31.240887000	192.168.1.182	209.84.3.254	HTTP	438	[TCP Retransmission] GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4921-Frag4
15:56:34.025778000	192.168.1.182	209.84.3.254	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4921-Frag492055 HTTP/1.1
15:56:34.797731000	209.84.3.254	192.168.1.182	HTTP	1160	HTTP/1.1 200 OK (video/f4f)
15:56:34.951920000	192.168.1.182	209.84.3.254	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4921-Frag492056 HTTP/1.1
15:56:35.423080000	209.84.3.254	192.168.1.182	HTTP	982	HTTP/1.1 200 OK (video/f4f)
15:56:35.627970000	192.168.1.182	209.84.3.254	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream2/stream2Seg4921-Frag492057 HTTP/1.1
15:56:36.015075000	209.84.3.254	192.168.1.182	HTTP	77	HTTP/1.1 200 OK (video/f4f)
15:56:39.594084000	192.168.1.182	209.84.3.254	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4921-Frag492058 HTTP/1.1
15:56:40.593164000	209.84.3.254	192.168.1.182	HTTP	584	HTTP/1.1 200 OK (video/f4f)
15:56:43.598280000	192.168.1.182	209.84.3.254	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4921-Frag492059 HTTP/1.1
15:56:44.390567000	209.84.3.254	192.168.1.182	HTTP	60	HTTP/1.1 200 OK (video/f4f)
15:56:47.629938000	192.168.1.182	209.84.3.254	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4921-Frag492060 HTTP/1.1
15:56:48.374911000	209.84.3.254	192.168.1.182	HTTP	1339	HTTP/1.1 200 OK (video/f4f)
15:56:51.591135000	192.168.1.182	209.84.3.254	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4921-Frag492061 HTTP/1.1

Figura 69. Captura 1 HDS para la propiedad de pruebas

En la captura se puede apreciar el periodo de corte y la recuperación de la conexión. Vemos como un fragmento se solicita con la calidad correspondiente al stream 2 pero rápidamente se vuelven a solicitar fragmentos de la calidad más alta.

Si seguimos aumentando la duración del corte, la reproducción se ve afectada durante mayor tiempo. Para un corte de 16 segundos, el vídeo cesa su emisión durante aproximadamente 9 segundos. Para un corte de 20 segundos, hay aproximadamente 13 segundos de interrupción del vídeo. Para este caso apreciamos que al reanudarse la reproducción, cambia el tamaño de la ventana (se hace más pequeña), como hemos comentado anteriormente.

Al provocar un corte de 24 segundos, la ausencia de vídeo aumenta hasta los 17 segundos aproximadamente. Hay que destacar que en algún caso (de las distintas pruebas que se hicieron) la reproducción de vídeo acababa parándose por completo, aunque por norma general se recuperaba el vídeo. Para un corte de 28 segundos, la reproducción se detenía durante casi ya 20 segundos. Por tanto, como es lógico, a mayor duración del corte causado, mayor es el tiempo en el que no tenemos vídeo.

Además comprobamos que hay 100 fragmentos asociados a un segmento (como se estableció en los parámetros de configuración de esta propiedad).

Time	Source	Destination	Protocol	Length	Info
16:32:11.781668000	192.168.1.182	8.26.222.254	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4926-Frag492592 HTTP/1.1
16:32:13.767154000	8.26.222.254	192.168.1.182	HTTP	382	HTTP/1.1 200 OK (video/f4f)
16:32:15.665978000	192.168.1.182	8.26.222.254	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4926-Frag492593 HTTP/1.1
16:32:17.260152000	8.26.222.254	192.168.1.182	HTTP	923	HTTP/1.1 200 OK (video/f4f)
16:32:19.666344000	192.168.1.182	8.26.222.254	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4926-Frag492594 HTTP/1.1
16:32:20.474617000	8.26.222.254	192.168.1.182	HTTP	253	HTTP/1.1 200 OK (video/f4f)
16:32:23.699374000	192.168.1.182	8.26.222.254	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4926-Frag492595 HTTP/1.1
16:32:24.477116000	8.26.222.254	192.168.1.182	HTTP	1205	HTTP/1.1 200 OK (video/f4f)
16:32:27.667012000	192.168.1.182	8.26.222.254	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4926-Frag492596 HTTP/1.1
16:32:28.396764000	8.26.222.254	192.168.1.182	HTTP	542	HTTP/1.1 200 OK (video/f4f)
16:32:31.665913000	192.168.1.182	8.26.222.254	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4926-Frag492597 HTTP/1.1
16:32:32.483925000	8.26.222.254	192.168.1.182	HTTP	563	HTTP/1.1 200 OK (video/f4f)
16:32:35.664673000	192.168.1.182	8.26.222.254	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4926-Frag492598 HTTP/1.1
16:32:36.438930000	8.26.222.254	192.168.1.182	HTTP	1045	HTTP/1.1 200 OK (video/f4f)
16:32:39.666238000	192.168.1.182	8.26.222.254	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4926-Frag492599 HTTP/1.1
16:32:40.819005000	8.26.222.254	192.168.1.182	HTTP	692	HTTP/1.1 200 OK (video/f4f)
16:32:43.664986000	192.168.1.182	8.26.222.254	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4926-Frag492600 HTTP/1.1
16:32:44.706533000	8.26.222.254	192.168.1.182	HTTP	1471	HTTP/1.1 200 OK (video/f4f)
16:32:47.782280000	192.168.1.182	8.254.202.254	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4927-Frag492601 HTTP/1.1
16:32:48.941205000	8.254.202.254	192.168.1.182	HTTP	1491	HTTP/1.1 200 OK (video/f4f)
16:32:51.663476000	192.168.1.182	8.254.202.254	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4927-Frag492602 HTTP/1.1
16:32:52.936743000	8.254.202.254	192.168.1.182	HTTP	592	HTTP/1.1 200 OK (video/f4f)
16:32:55.665787000	192.168.1.182	8.254.202.254	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4927-Frag492603 HTTP/1.1

Figura 70. Captura 2 HDS para la propiedad de pruebas

3.3.3.3. Pruebas HDS cambiando la duración del segmento

Para HDS vamos a utilizar la misma propiedad que anteriormente solicitamos para pruebas al proveedor de la CDN que es la que nos permite modificar el parámetro que nos interesa. En este caso vamos a variar la duración de los fragmentos que van a pasar de 4 segundos a 8 segundos. La duración de los segmentos la mantenemos en 400 segundos, lo que va a suponer que tendremos un total de 50 fragmentos por segmento.

Para cambiar la duración de los fragmentos accedemos al AOM, y una vez parada la publicación de la propiedad en el codificador, podemos modificar el parámetro en cuestión. Posteriormente, pasamos a activar de nuevo la publicación de la propiedad en el codificador.

Los primeros cambios que observamos son que ahora los fragmentos se envían cada 8 segundos, además, el proceso de inicialización del buffer es más lento.

Time	Source	Destination	Protocol	Length	Info
17:04:29.660785000	192.168.1.182	192.221.126.25	HTTP	447	GET /live/abertis-hdstest/_definst_/live.f4m HTTP/1.1
17:04:29.750772000	192.221.126.25	192.168.1.182	HTTP/XML	78	HTTP/1.1 200 OK
17:04:29.761587000	192.168.1.182	192.221.126.25	HTTP	480	GET /live/streams/abertis-hdstest/streams/_definst_/stream2/stream2.bootstrap HTTP/1.1
17:04:29.802749000	192.168.1.182	192.221.126.25	HTTP	480	GET /live/streams/abertis-hdstest/streams/_definst_/stream1/stream1.bootstrap HTTP/1.1
17:04:29.803379000	192.168.1.182	192.221.126.25	HTTP	480	GET /live/streams/abertis-hdstest/streams/_definst_/stream3/stream3.bootstrap HTTP/1.1
17:04:29.803992000	192.168.1.182	192.221.126.25	HTTP	480	GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4.bootstrap HTTP/1.1
17:04:29.830620000	192.221.126.25	192.168.1.182	HTTP	168	HTTP/1.1 200 OK (application/binary)
17:04:29.869404000	192.221.126.25	192.168.1.182	HTTP	168	HTTP/1.1 200 OK (application/binary)
17:04:29.878542000	192.221.126.25	192.168.1.182	HTTP	168	HTTP/1.1 200 OK (application/binary)
17:04:29.881277000	192.221.126.25	192.168.1.182	HTTP	476	HTTP/1.1 200 OK (application/binary)
17:04:30.511752000	192.168.1.182	192.221.126.25	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream2/stream2Seg4931-Frag246537 HTTP/1.1
17:04:31.144760000	192.221.126.25	192.168.1.182	HTTP	1211	HTTP/1.1 200 OK (video/f4f)
17:04:32.709596000	192.168.1.182	192.221.126.25	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4931-Frag246538 HTTP/1.1
17:04:33.894390000	192.221.126.25	192.168.1.182	HTTP	531	HTTP/1.1 200 OK (video/f4f)
17:04:37.263393000	192.168.1.182	192.221.126.25	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4931-Frag246539 HTTP/1.1
17:04:38.899100000	192.221.126.25	192.168.1.182	HTTP	1037	HTTP/1.1 200 OK (video/f4f)
17:04:44.594424000	192.168.1.182	192.221.126.25	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4931-Frag246540 HTTP/1.1
17:04:46.309123000	192.221.126.25	192.168.1.182	HTTP	119	HTTP/1.1 200 OK (video/f4f)
17:04:52.590398000	192.168.1.182	192.221.126.25	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4931-Frag246541 HTTP/1.1
17:04:53.936559000	192.221.126.25	192.168.1.182	HTTP	709	HTTP/1.1 200 OK (video/f4f)

Figura 71. Captura del inicio de HDS cambiando la duración del fragmento

En la captura realizada se puede ver que al inicio de la reproducción se solicitan 3 fragmentos en un periodo de unos 7 segundos, para posteriormente ya comenzar a recibir fragmentos cada 8 segundos.

También podemos ver que la capacidad del buffer aumenta y llega a alcanzar los 14.4 segundos aproximadamente. El buffer va variando entre los 7.7 segundos y los 14.4 segundos durante una transmisión de vídeo continua.

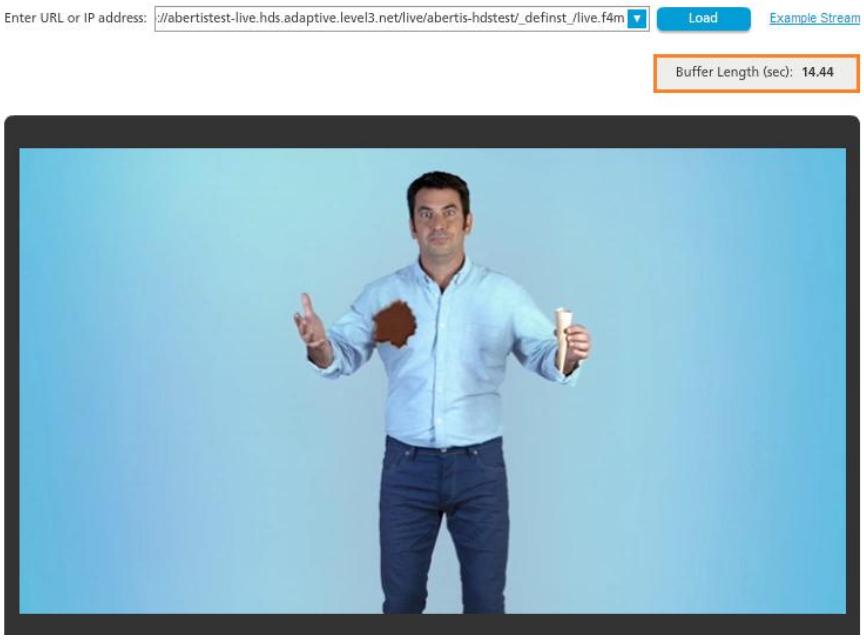
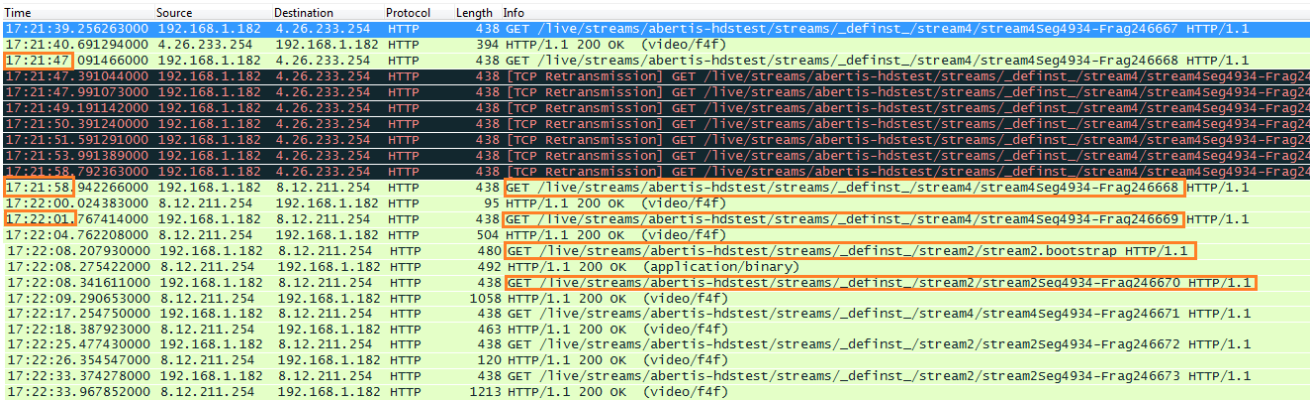


Figura 72. Comportamiento del buffer cambiando la duración del fragmento en HDS

Si realizamos un corte prolongado con el dispositivo Albedo vemos que la reproducción continúa unos 13/14 segundos como máximo (lo que equivaldría aproximadamente a unos 2 fragmentos de duración 8 segundos). De nuevo vemos el mismo efecto que anteriormente, a raíz del corte el valor del buffer va disminuyendo hasta alcanzar el mínimo (cero) que es cuando se para el vídeo.

En la batería de pruebas realizadas vimos que a pesar de introducir un corte bastante largo (50 segundos) casi siempre se acababa recuperando la reproducción.

Como en casos anteriores, procedimos a ir aumentando la duración de los cortes. Provocando un corte de 8 segundos, veíamos como no se producía ninguna afectación cara al usuario. Al introducir un corte de 16 segundos, desde las 17:21:41 hasta las 17:21:57, la señal se corta durante unos 5/6 segundos. Hay que destacar que en función del momento en el que iniciemos el corte (la duración que almacena el buffer va variando), la afectación es mayor o menor.



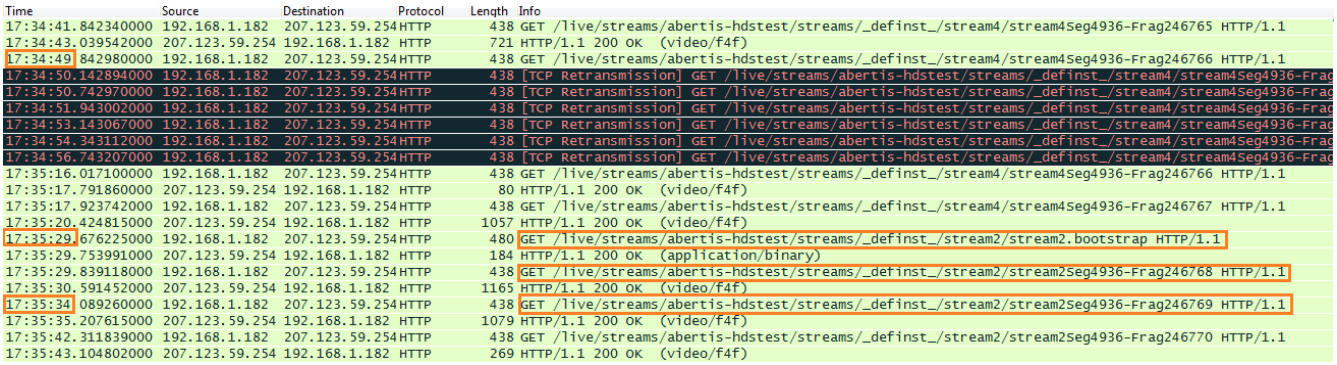
Time	Source	Destination	Protocol	Length	Info
17:21:39.256263000	192.168.1.182	4.26.233.254	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4934-Frag246667 HTTP/1.1
17:21:40.691294000	4.26.233.254	192.168.1.182	HTTP	394	HTTP/1.1 200 OK (video/f4f)
17:21:47.091466000	192.168.1.182	4.26.233.254	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4934-Frag246668 HTTP/1.1
17:21:47.391044000	192.168.1.182	4.26.233.254	HTTP	438	[TCP Retransmission] GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4934-Frag24
17:21:47.991073000	192.168.1.182	4.26.233.254	HTTP	438	[TCP Retransmission] GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4934-Frag24
17:21:49.191142000	192.168.1.182	4.26.233.254	HTTP	438	[TCP Retransmission] GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4934-Frag24
17:21:50.391240000	192.168.1.182	4.26.233.254	HTTP	438	[TCP Retransmission] GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4934-Frag24
17:21:51.591291000	192.168.1.182	4.26.233.254	HTTP	438	[TCP Retransmission] GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4934-Frag24
17:21:53.991389000	192.168.1.182	4.26.233.254	HTTP	438	[TCP Retransmission] GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4934-Frag24
17:21:58.792363000	192.168.1.182	4.26.233.254	HTTP	438	[TCP Retransmission] GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4934-Frag24
17:21:58.942266000	192.168.1.182	8.12.211.254	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4934-Frag246668 HTTP/1.1
17:22:00.024383000	8.12.211.254	192.168.1.182	HTTP	95	HTTP/1.1 200 OK (video/f4f)
17:22:01.767414000	192.168.1.182	8.12.211.254	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4934-Frag246669 HTTP/1.1
17:22:04.762208000	8.12.211.254	192.168.1.182	HTTP	504	HTTP/1.1 200 OK (video/f4f)
17:22:08.207930000	192.168.1.182	8.12.211.254	HTTP	480	GET /live/streams/abertis-hdstest/streams/_definst_/stream2/stream2.bootstrap HTTP/1.1
17:22:08.275422000	8.12.211.254	192.168.1.182	HTTP	492	HTTP/1.1 200 OK (application/binary)
17:22:08.341611000	192.168.1.182	8.12.211.254	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream2/stream2Seg4934-Frag246670 HTTP/1.1
17:22:09.290653000	8.12.211.254	192.168.1.182	HTTP	1058	HTTP/1.1 200 OK (video/f4f)
17:22:17.254750000	192.168.1.182	8.12.211.254	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4934-Frag246671 HTTP/1.1
17:22:18.387923000	8.12.211.254	192.168.1.182	HTTP	463	HTTP/1.1 200 OK (video/f4f)
17:22:25.477430000	192.168.1.182	8.12.211.254	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream2/stream2Seg4934-Frag246672 HTTP/1.1
17:22:26.354547000	8.12.211.254	192.168.1.182	HTTP	120	HTTP/1.1 200 OK (video/f4f)
17:22:33.374278000	192.168.1.182	8.12.211.254	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream2/stream2Seg4934-Frag246673 HTTP/1.1
17:22:33.967852000	8.12.211.254	192.168.1.182	HTTP	1213	HTTP/1.1 200 OK (video/f4f)

Figura 73. Captura 2 HDS cambiando la duración del fragmento

En la captura del corte de 16 segundos, vemos que al recuperar la conexión se solicitan dos fragmentos en apenas 3 segundos. Además, hay un cambio de calidad (del stream 4 al stream 2).

Al provocar un corte de 24 segundos, se produce un corte en la reproducción de aproximadamente unos 15 segundos. Si aumentamos a 28 segundos, el corte se extiende hasta unos 17 segundos. Las capturas que realizamos mostraban un comportamiento similar al caso de 16 segundos, con una petición de fragmentos en poco espacio de tiempo al recuperar la conexión y con un cambio de calidad.

Por último, al provocar un corte de 32 segundos, desde las 17:34:44 hasta las 17:35:16, la reproducción se paraba durante unos 20 segundos.



Time	Source	Destination	Protocol	Length	Info
17:34:41.842340000	192.168.1.182	207.123.59.254	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4936-Frag246765 HTTP/1.1
17:34:43.039542000	207.123.59.254	192.168.1.182	HTTP	721	HTTP/1.1 200 OK (video/f4f)
17:34:49.842980000	192.168.1.182	207.123.59.254	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4936-Frag246766 HTTP/1.1
17:34:50.142894000	192.168.1.182	207.123.59.254	HTTP	438	[TCP Retransmission] GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4936-Frag
17:34:50.742970000	192.168.1.182	207.123.59.254	HTTP	438	[TCP Retransmission] GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4936-Frag
17:34:51.943002000	192.168.1.182	207.123.59.254	HTTP	438	[TCP Retransmission] GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4936-Frag
17:34:53.143067000	192.168.1.182	207.123.59.254	HTTP	438	[TCP Retransmission] GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4936-Frag
17:34:54.343112000	192.168.1.182	207.123.59.254	HTTP	438	[TCP Retransmission] GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4936-Frag
17:34:56.743207000	192.168.1.182	207.123.59.254	HTTP	438	[TCP Retransmission] GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4936-Frag
17:35:16.017100000	192.168.1.182	207.123.59.254	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4936-Frag246766 HTTP/1.1
17:35:17.791860000	207.123.59.254	192.168.1.182	HTTP	80	HTTP/1.1 200 OK (video/f4f)
17:35:17.923742000	192.168.1.182	207.123.59.254	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream4/stream4Seg4936-Frag246767 HTTP/1.1
17:35:20.424815000	207.123.59.254	192.168.1.182	HTTP	1057	HTTP/1.1 200 OK (video/f4f)
17:35:29.676225000	192.168.1.182	207.123.59.254	HTTP	480	GET /live/streams/abertis-hdstest/streams/_definst_/stream2/stream2.bootstrap HTTP/1.1
17:35:29.753991000	207.123.59.254	192.168.1.182	HTTP	184	HTTP/1.1 200 OK (application/binary)
17:35:29.839118000	192.168.1.182	207.123.59.254	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream2/stream2Seg4936-Frag246768 HTTP/1.1
17:35:30.591452000	207.123.59.254	192.168.1.182	HTTP	1165	HTTP/1.1 200 OK (video/f4f)
17:35:34.089260000	192.168.1.182	207.123.59.254	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream2/stream2Seg4936-Frag246769 HTTP/1.1
17:35:35.207615000	207.123.59.254	192.168.1.182	HTTP	1079	HTTP/1.1 200 OK (video/f4f)
17:35:42.311839000	192.168.1.182	207.123.59.254	HTTP	438	GET /live/streams/abertis-hdstest/streams/_definst_/stream2/stream2Seg4936-Frag246770 HTTP/1.1
17:35:43.104802000	207.123.59.254	192.168.1.182	HTTP	269	HTTP/1.1 200 OK (video/f4f)

Figura 74. Captura 3 HDS cambiando la duración del fragmento

En la última prueba podemos ver que se baja de nuevo a la calidad más baja (con el consiguiente cambio en la resolución de la pantalla). En apenas 5 segundos se piden 3 fragmentos de la calidad correspondiente al stream 2.

HDS			
PLATAFORMA/PLAYER	**TAMAÑO DEL BUFFER**	**DURACIÓN CHUNKS**	**MAXIMO CORTE SOPORTADO**
Nubeox	6 segundos	4 segundos	+ 50 segundos
Reproductor CDN	7.7-12 segundos	4 segundos	+28 segundos
Reproductor CDN	7.7-14 segundos	8 segundos	+ 40 segundos

Tabla 7. Resumen Pruebas HDS

Analizando los resultados de las pruebas llevadas a cabo con el protocolo HDS, podemos ver que el reproductor del proveedor de servicios de la CDN tiene mayor capacidad de buffer que el de la aplicación de Nubeox.

Además, al variar la duración del fragmento, la capacidad del buffer se ha visto incrementada ligeramente.

También podemos ver que ante cortes de larga duración, a pesar de provocar interrupciones en la señal de vídeo reproducida en el player, el reproductor es capaz de reanudar el contenido que ofrece sin tener que recurrir a hacerlo de forma manual.

3.3.4. Pruebas protocolo MPEG-DASH

MPEG-DASH es el protocolo más novedoso y aún se encuentra en fase de evolución. Presenta la ventaja de que se puede utilizar en cualquier dispositivo y se puede utilizar cualquier técnica de DRM para la encriptación. Esto a su vez podría ser un inconveniente ya que si el DRM empleado es débil, será fácilmente descifrable.

Para realizar las pruebas vamos a utilizar una TV conectada (marca Panasonic) utilizando una aplicación para *smart* TVs denominada "Web Browser". Esta aplicación utiliza un reproductor de MPEG-DASH.

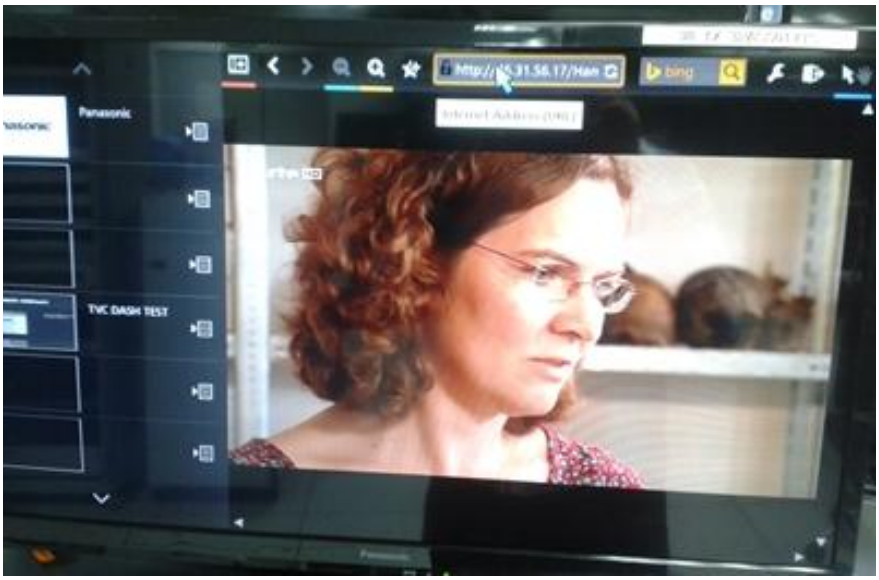


Figura 75. Reproductor MPEG-DASH en TV Panasonic

Utilizaremos una propiedad generada en uno de los equipos (codificadores) Harmonic del Laboratorio. El escenario para las pruebas de MPEG-DASH estará formado por dos dispositivos Albedo, el Net.Storm que hemos venido utilizando para provocar los cortes y otro dispositivo similar, el Net.Shark [34] que nos va a servir de elemento puente entre la TV y el Net.Storm y que nos permitirá además poder capturar el tráfico con el ordenador portátil.

El escenario en este caso varía ligeramente respecto a los escenarios utilizados para los tres protocolos anteriores, aunque el objetivo es el mismo. Nos apoyamos en el dispositivo Net.Shark con el único fin de poder capturar el tráfico entre la TV (el reproductor empleado para MPEG-DASH) y la CDN. Por tanto, la única diferencia es la utilización de un dispositivo extra que nos permita realizar dicha monitorización. Otra opción podría haber sido emplear un escenario similar al utilizado en HLS, pero debido a algunos problemas de firewall (la propiedad a utilizar para MPEG-DASH era de pruebas y estaba en un equipo del laboratorio) que nos limitaban la conectividad vía Wifi, se decidió emplear el escenario de la figura 76.

Figura 76. Escenario de MPEG-DASH

En el protocolo MPEG-DASH el fichero .mpd (*Media Presentation Description*) juega el papel del archivo .m3u8 de HLS o del archivo de bootstrap de HDS, es decir, es el Manifest de MPEG-DASH.

En nuestro caso la propiedad de pruebas que vamos a utilizar tiene una URL pública definida:

http://46.31.56.17/Harmonic/DASH/testmulti/test.mpd

Accediendo al fichero .mpd podemos ver que la duración de los segmentos es de 5 segundos y las diferentes calidades que tenemos del stream. En total tenemos 4 calidades de vídeo y una de audio.

```
<?xml version="1.0" encoding="UTF-8"?>
- <MPD xsi:schemaLocation="urn:mpeg:DASH:schema:MPD:2011 DASH-MPD.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="urn:mpeg:DASH:schema:MPD:2011" timeShiftBufferDepth="PT5M" minBufferTime="PT1S" minimumUpdatePeriod="PT5S" availabilityStartTime="2014-06-
25T08:47:02Z" type="dynamic" profiles="urn:mpeg:dash:profile:isoff-live:2011">
  - <Period start="PT0S">
    - <AdaptationSet segmentAlignment="true" startWithSAP="1" mimeType="video/mp4">
        <SegmentTemplate initialization="$RepresentationID$_init.m4i" media="$RepresentationID$_Segment-$Number$.m4v" startNumber="1" duration="5"
          presentationTimeOffset="218"/>
        <Representation bandwidth="1080000" id="Stream_2_1080000" scanType="progressive" codecs="avc1.64001f" frameRate="25" height="576" width="1024"/>
        <Representation bandwidth="3240000" id="Stream_1_3240000" scanType="progressive" codecs="avc1.64001f" frameRate="25" height="720" width="1280"/>
        <Representation bandwidth="5400000" id="Stream_0_5400000" scanType="progressive" codecs="avc1.64001f" frameRate="25" height="720" width="1280"/>
        <Representation bandwidth="540000" id="Stream_3_540000" scanType="progressive" codecs="avc1.64001f" frameRate="25" height="360" width="640"/>
      </AdaptationSet>
    - <AdaptationSet lang="deu" segmentAlignment="true" startWithSAP="1" mimeType="audio/mp4">
        <SegmentTemplate initialization="$RepresentationID$_init.m4i" media="$RepresentationID$_Segment-$Number$.m4a" startNumber="1" duration="5"
          presentationTimeOffset="218"/>
        <Representation bandwidth="192000" id="Stream_4_192000" codecs="mp4a.40.2" audioSamplingRate="48000"/>
      </AdaptationSet>
  </Period>
</MPD>
```

Figura 77. Archivo .mpd de MPEG-DASH

Como se puede ver en la figura anterior, tenemos un set adaptativo dónde se definen las distintas representaciones/calidades de vídeo. Cada una tiene su propio identificador "Stream_*i*", donde *i* = 0, 1, 2, 3. Por otro lado tenemos otro set adaptativo donde se encuadra la calidad de audio. Ambos sets adaptativos se engloban en un periodo siguiendo la misma estructura que vimos en la parte teórica de este protocolo.

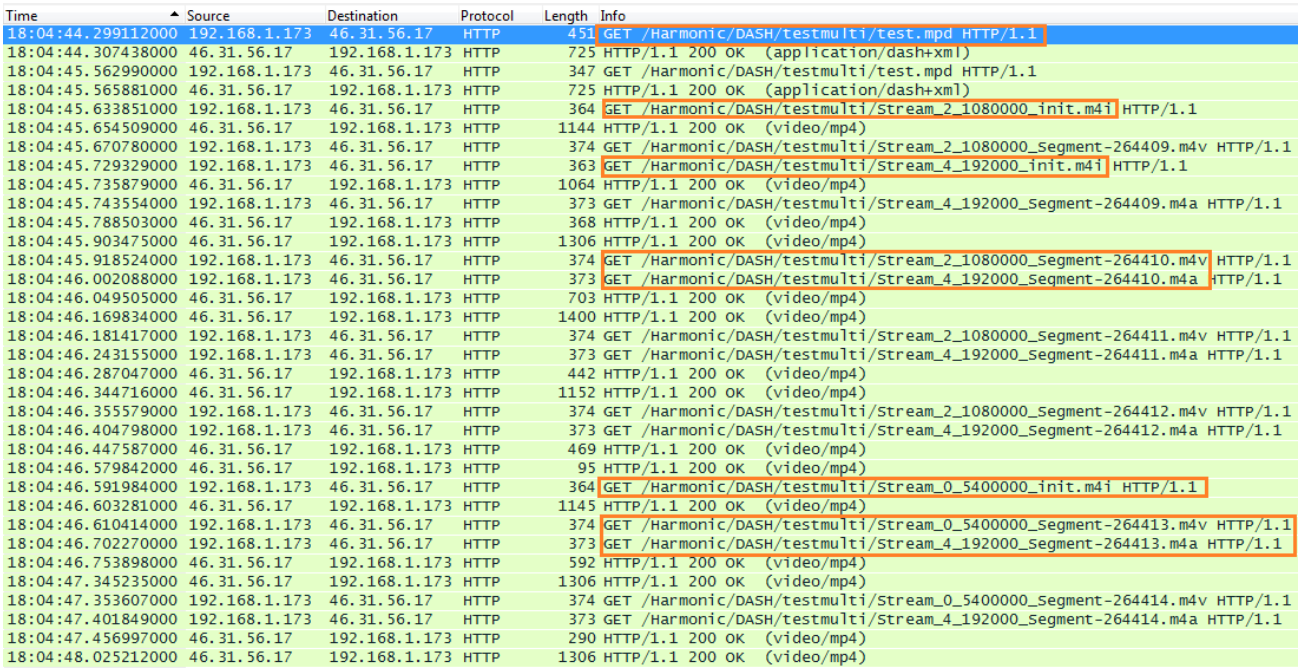
3.3.4.1. Pruebas MPEG-DASH con TV conectada

Empleando la propiedad anterior y con el escenario descrito, vamos a presentar una batería de pruebas con el protocolo MPEG-DASH para ver su comportamiento y analizar su robustez.

El DVR está configurado con un valor de 300 segundos (5 minutos) en esta propiedad.

Figura 78. Montaje en el laboratorio de los dispositivos Albedo

En primer lugar realizamos una captura de tráfico que recoja el inicio de la reproducción y el envío de segmentos de audio y vídeo. Vamos a ver que los segmentos de audio y vídeo se solicitan por separado como ocurría con los fragmentos en Smooth Streaming. Además veremos que los números de los segmentos van aumentando de forma secuencial.



Time	Source	Destination	Protocol	Length	Info
18:04:44.299112000	192.168.1.173	46.31.56.17	HTTP	451	GET /Harmonic/DASH/testmulti/test.mpd HTTP/1.1
18:04:44.307438000	46.31.56.17	192.168.1.173	HTTP	725	HTTP/1.1 200 OK (application/dash+xml)
18:04:45.562990000	192.168.1.173	46.31.56.17	HTTP	347	GET /Harmonic/DASH/testmulti/test.mpd HTTP/1.1
18:04:45.565881000	46.31.56.17	192.168.1.173	HTTP	725	HTTP/1.1 200 OK (application/dash+xml)
18:04:45.633851000	192.168.1.173	46.31.56.17	HTTP	364	GET /Harmonic/DASH/testmulti/Stream_2_1080000_init.m4i HTTP/1.1
18:04:45.654509000	46.31.56.17	192.168.1.173	HTTP	1144	HTTP/1.1 200 OK (video/mp4)
18:04:45.670780000	192.168.1.173	46.31.56.17	HTTP	374	GET /Harmonic/DASH/testmulti/Stream_2_1080000_Segment-264409.m4v HTTP/1.1
18:04:45.729329000	192.168.1.173	46.31.56.17	HTTP	363	GET /Harmonic/DASH/testmulti/Stream_4_192000_init.m4i HTTP/1.1
18:04:45.735879000	46.31.56.17	192.168.1.173	HTTP	1064	HTTP/1.1 200 OK (video/mp4)
18:04:45.743554000	192.168.1.173	46.31.56.17	HTTP	373	GET /Harmonic/DASH/testmulti/Stream_4_192000_Segment-264409.m4a HTTP/1.1
18:04:45.788503000	46.31.56.17	192.168.1.173	HTTP	368	HTTP/1.1 200 OK (video/mp4)
18:04:45.903475000	46.31.56.17	192.168.1.173	HTTP	1306	HTTP/1.1 200 OK (video/mp4)
18:04:45.918524000	192.168.1.173	46.31.56.17	HTTP	374	GET /Harmonic/DASH/testmulti/Stream_2_1080000_Segment-264410.m4v HTTP/1.1
18:04:46.002088000	192.168.1.173	46.31.56.17	HTTP	373	GET /Harmonic/DASH/testmulti/Stream_4_192000_Segment-264410.m4a HTTP/1.1
18:04:46.049505000	46.31.56.17	192.168.1.173	HTTP	703	HTTP/1.1 200 OK (video/mp4)
18:04:46.169834000	46.31.56.17	192.168.1.173	HTTP	1400	HTTP/1.1 200 OK (video/mp4)
18:04:46.181417000	192.168.1.173	46.31.56.17	HTTP	374	GET /Harmonic/DASH/testmulti/Stream_2_1080000_Segment-264411.m4v HTTP/1.1
18:04:46.243155000	192.168.1.173	46.31.56.17	HTTP	373	GET /Harmonic/DASH/testmulti/Stream_4_192000_Segment-264411.m4a HTTP/1.1
18:04:46.287047000	46.31.56.17	192.168.1.173	HTTP	442	HTTP/1.1 200 OK (video/mp4)
18:04:46.344716000	46.31.56.17	192.168.1.173	HTTP	1152	HTTP/1.1 200 OK (video/mp4)
18:04:46.355579000	192.168.1.173	46.31.56.17	HTTP	374	GET /Harmonic/DASH/testmulti/Stream_2_1080000_Segment-264412.m4v HTTP/1.1
18:04:46.404798000	192.168.1.173	46.31.56.17	HTTP	373	GET /Harmonic/DASH/testmulti/Stream_4_192000_Segment-264412.m4a HTTP/1.1
18:04:46.447587000	46.31.56.17	192.168.1.173	HTTP	469	HTTP/1.1 200 OK (video/mp4)
18:04:46.579842000	46.31.56.17	192.168.1.173	HTTP	95	HTTP/1.1 200 OK (video/mp4)
18:04:46.591984000	192.168.1.173	46.31.56.17	HTTP	364	GET /Harmonic/DASH/testmulti/Stream_0_5400000_init.m4i HTTP/1.1
18:04:46.603281000	46.31.56.17	192.168.1.173	HTTP	1145	HTTP/1.1 200 OK (video/mp4)
18:04:46.610414000	192.168.1.173	46.31.56.17	HTTP	374	GET /Harmonic/DASH/testmulti/Stream_0_5400000_Segment-264413.m4v HTTP/1.1
18:04:46.702270000	192.168.1.173	46.31.56.17	HTTP	373	GET /Harmonic/DASH/testmulti/Stream_4_192000_Segment-264413.m4a HTTP/1.1
18:04:46.753898000	46.31.56.17	192.168.1.173	HTTP	592	HTTP/1.1 200 OK (video/mp4)
18:04:47.345235000	46.31.56.17	192.168.1.173	HTTP	1306	HTTP/1.1 200 OK (video/mp4)
18:04:47.353607000	192.168.1.173	46.31.56.17	HTTP	374	GET /Harmonic/DASH/testmulti/Stream_0_5400000_Segment-264414.m4v HTTP/1.1
18:04:47.401849000	192.168.1.173	46.31.56.17	HTTP	373	GET /Harmonic/DASH/testmulti/Stream_4_192000_Segment-264414.m4a HTTP/1.1
18:04:47.456997000	46.31.56.17	192.168.1.173	HTTP	290	HTTP/1.1 200 OK (video/mp4)
18:04:48.025212000	46.31.56.17	192.168.1.173	HTTP	1306	HTTP/1.1 200 OK (video/mp4)

Figura 79. Captura 1 del inicio de MPEG-DASH

Al comienzo los segmentos se envían en un corto espacio de tiempo, mientras se va llenando el buffer. Cuando la reproducción está estable los segmentos se mandan cada 5 segundos (duración de los mismos).

En primer lugar se solicita el fichero .mpd que tiene el formato que hemos visto anteriormente. A continuación se piden los segmentos de inicialización de cada representación, vemos que se pide la correspondiente al Stream_2 que corresponde a la segunda calidad más baja. Por separado se pide el segmento de inicialización del audio (estos archivos se definen en un formato .m4i). Y a partir de ahí se solicitan los segmentos de audio y vídeo (.m4a y .m4v). En la figura anterior podemos apreciar un cambio de calidad (a la más alta) cuando se solicita el fichero de inicialización correspondiente al Stream_0.

La primera prueba que realizamos es provocar un corte muy largo, es decir, una pérdida de conexión para ver cuánto tiempo a partir del corte el reproductor continúa funcionando.

Se reproducen 25 segundos de vídeo a partir del corte, lo que equivale a 5 segmentos de vídeo teniendo en cuenta que cada segmento tiene una duración de 5 segundos. Por tanto en el buffer teníamos guardados 5 segmentos para reproducir a partir del corte.

Cuando cesa el efecto del corte, el reproductor ya no es capaz de continuar con la reproducción teniendo el usuario que volver a iniciar manualmente la misma.

A continuación observamos el efecto que se provoca en la aplicación de MPEG-DASH al realizar cortes variando la duración de los mismos, aumentando de 5 en 5 segundos.

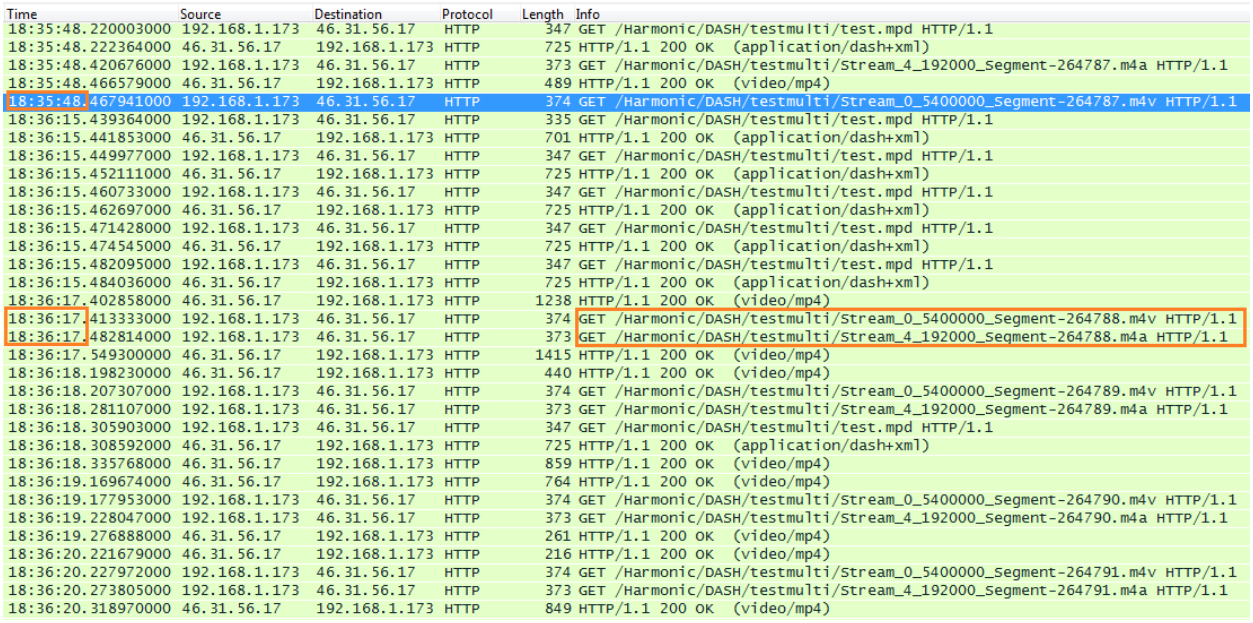
Al introducir un corte de 5 segundos, desde las 18:22:40 a las 18:22:45, en el reproductor no apreciamos ningún efecto anómalo, como es lógico ya que hay segmentos en el buffer para continuar la reproducción. En la captura de la figura mostrada a continuación, podemos ver que antes del corte los segmentos se estaban solicitando cada 5 segundos. Nada más recuperarse del corte, se solicita el archivo .mpd y se piden varios segmentos de audio y vídeo en un corto espacio de tiempo de apenas 1 segundo. Enseguida se recupera la estabilidad y se vuelve a observar el tráfico habitual.

Time	Source	Destination	Protocol	Length	Info
18:22:20.313815000	192.168.1.173	46.31.56.17	HTTP	347	GET /Harmonic/DASH/testmulti/test.mpd HTTP/1.1
18:22:20.316297000	46.31.56.17	192.168.1.173	HTTP	725	HTTP/1.1 200 OK (application/dash+xml)
18:22:20.691740000	192.168.1.173	46.31.56.17	HTTP	374	GET /Harmonic/DASH/testmulti/Stream_0_5400000_Segment-264625.m4v HTTP/1.1
18:22:20.745745000	192.168.1.173	46.31.56.17	HTTP	373	GET /Harmonic/DASH/testmulti/Stream_4_192000_Segment-264625.m4a HTTP/1.1
18:22:20.842529000	46.31.56.17	192.168.1.173	HTTP	327	HTTP/1.1 200 OK (video/mp4)
18:22:25.232217000	192.168.1.173	46.31.56.17	HTTP	347	GET /Harmonic/DASH/testmulti/test.mpd HTTP/1.1
18:22:25.234441000	46.31.56.17	192.168.1.173	HTTP	725	HTTP/1.1 200 OK (application/dash+xml)
18:22:25.254202000	46.31.56.17	192.168.1.173	HTTP	664	HTTP/1.1 200 OK (video/mp4)
18:22:25.574542000	192.168.1.173	46.31.56.17	HTTP	373	GET /Harmonic/DASH/testmulti/Stream_4_192000_Segment-264626.m4a HTTP/1.1
18:22:25.613610000	46.31.56.17	192.168.1.173	HTTP	807	HTTP/1.1 200 OK (video/mp4)
18:22:25.615871000	192.168.1.173	46.31.56.17	HTTP	374	GET /Harmonic/DASH/testmulti/Stream_0_5400000_Segment-264626.m4v HTTP/1.1
18:22:30.412713000	192.168.1.173	46.31.56.17	HTTP	347	GET /Harmonic/DASH/testmulti/test.mpd HTTP/1.1
18:22:30.414883000	46.31.56.17	192.168.1.173	HTTP	725	HTTP/1.1 200 OK (application/dash+xml)
18:22:31.092510000	46.31.56.17	192.168.1.173	HTTP	1339	HTTP/1.1 200 OK (video/mp4)
18:22:31.373902000	192.168.1.173	46.31.56.17	HTTP	374	GET /Harmonic/DASH/testmulti/Stream_0_5400000_Segment-264627.m4v HTTP/1.1
18:22:31.427909000	192.168.1.173	46.31.56.17	HTTP	373	GET /Harmonic/DASH/testmulti/Stream_4_192000_Segment-264627.m4a HTTP/1.1
18:22:31.491440000	46.31.56.17	192.168.1.173	HTTP	396	HTTP/1.1 200 OK (video/mp4)
18:22:35.096155000	46.31.56.17	192.168.1.173	HTTP	460	HTTP/1.1 200 OK (video/mp4)
18:22:35.342945000	192.168.1.173	46.31.56.17	HTTP	347	GET /Harmonic/DASH/testmulti/test.mpd HTTP/1.1
18:22:35.345432000	46.31.56.17	192.168.1.173	HTTP	725	HTTP/1.1 200 OK (application/dash+xml)
18:22:35.415571000	192.168.1.173	46.31.56.17	HTTP	374	GET /Harmonic/DASH/testmulti/Stream_0_5400000_Segment-264628.m4v HTTP/1.1
18:22:35.487715000	192.168.1.173	46.31.56.17	HTTP	373	GET /Harmonic/DASH/testmulti/Stream_4_192000_Segment-264628.m4a HTTP/1.1
18:22:35.534304000	46.31.56.17	192.168.1.173	HTTP	328	HTTP/1.1 200 OK (video/mp4)
18:22:46.396200000	192.168.1.173	46.31.56.17	HTTP	347	GET /Harmonic/DASH/testmulti/test.mpd HTTP/1.1
18:22:46.398537000	46.31.56.17	192.168.1.173	HTTP	725	HTTP/1.1 200 OK (application/dash+xml)
18:22:46.410897000	192.168.1.173	46.31.56.17	HTTP	347	GET /Harmonic/DASH/testmulti/test.mpd HTTP/1.1
18:22:46.412862000	46.31.56.17	192.168.1.173	HTTP	725	HTTP/1.1 200 OK (application/dash+xml)
18:22:49.295464000	46.31.56.17	192.168.1.173	HTTP	390	HTTP/1.1 200 OK (video/mp4)
18:22:49.302460000	192.168.1.173	46.31.56.17	HTTP	373	GET /Harmonic/DASH/testmulti/Stream_4_192000_Segment-264629.m4a HTTP/1.1
18:22:49.344773000	46.31.56.17	192.168.1.173	HTTP	822	HTTP/1.1 200 OK (video/mp4)
18:22:49.345441000	192.168.1.173	46.31.56.17	HTTP	374	GET /Harmonic/DASH/testmulti/Stream_0_5400000_Segment-264629.m4v HTTP/1.1
18:22:50.220848000	46.31.56.17	192.168.1.173	HTTP	447	HTTP/1.1 200 OK (video/mp4)
18:22:50.228448000	192.168.1.173	46.31.56.17	HTTP	374	GET /Harmonic/DASH/testmulti/Stream_0_5400000_Segment-264630.m4v HTTP/1.1
18:22:50.242934000	192.168.1.173	46.31.56.17	HTTP	347	GET /Harmonic/DASH/testmulti/test.mpd HTTP/1.1
18:22:50.245140000	46.31.56.17	192.168.1.173	HTTP	725	HTTP/1.1 200 OK (application/dash+xml)
18:22:50.283156000	192.168.1.173	46.31.56.17	HTTP	373	GET /Harmonic/DASH/testmulti/Stream_4_192000_Segment-264630.m4a HTTP/1.1
18:22:50.330958000	46.31.56.17	192.168.1.173	HTTP	162	HTTP/1.1 200 OK (video/mp4)
18:22:51.087315000	46.31.56.17	192.168.1.173	HTTP	728	HTTP/1.1 200 OK (video/mp4)
18:22:51.095426000	192.168.1.173	46.31.56.17	HTTP	374	GET /Harmonic/DASH/testmulti/Stream_0_5400000_Segment-264631.m4v HTTP/1.1
18:22:51.146883000	192.168.1.173	46.31.56.17	HTTP	373	GET /Harmonic/DASH/testmulti/Stream_4_192000_Segment-264631.m4a HTTP/1.1
18:22:51.200625000	46.31.56.17	192.168.1.173	HTTP	965	HTTP/1.1 200 OK (video/mp4)

Figura 80. Captura 2 MPEG-DASH con propiedad de pruebas

Al provocar un corte de 10 segundos observamos una situación parecida, en el reproductor no se aprecia ningún efecto. La única diferencia radica en que se pide algún número de segmentos más en un espacio de tiempo inferior a 5 segundos (antes de recuperar la normalidad). Ocurre lo mismo para cortes de 15 y 20 segundos. En el primer caso se piden hasta 5 segmentos de vídeo en 6 segundos y en el corte de 20 segundos se piden 7 segmentos en 8 segundos.

Al provocar un corte de 25 segundos, desde las 18:35:50 a las 18:36:15, ya tenemos un pequeño corte en el reproductor de MPEG-DASH de la TV de 1/2 segundos de duración. En la captura podemos ver que desde que se recupera la conexión, el player realiza varias solicitudes del "Manifest" o fichero .mpd antes de pedir y recibir los segmentos de audio y vídeo.



Time	Source	Destination	Protocol	Length	Info
18:35:48.220003000	192.168.1.173	46.31.56.17	HTTP	347	GET /Harmonic/DASH/testmulti/test.mpd HTTP/1.1
18:35:48.222364000	46.31.56.17	192.168.1.173	HTTP	725	HTTP/1.1 200 OK (application/dash+xml)
18:35:48.420676000	192.168.1.173	46.31.56.17	HTTP	373	GET /Harmonic/DASH/testmulti/Stream_4_192000_Segment-264787.m4a HTTP/1.1
18:35:48.466579000	46.31.56.17	192.168.1.173	HTTP	489	HTTP/1.1 200 OK (video/mp4)
18:35:48.467941000	192.168.1.173	46.31.56.17	HTTP	374	GET /Harmonic/DASH/testmulti/Stream_0_5400000_Segment-264787.m4v HTTP/1.1
18:36:15.439364000	192.168.1.173	46.31.56.17	HTTP	335	GET /Harmonic/DASH/testmulti/test.mpd HTTP/1.1
18:36:15.441853000	46.31.56.17	192.168.1.173	HTTP	701	HTTP/1.1 200 OK (application/dash+xml)
18:36:15.449977000	192.168.1.173	46.31.56.17	HTTP	347	GET /Harmonic/DASH/testmulti/test.mpd HTTP/1.1
18:36:15.452111000	46.31.56.17	192.168.1.173	HTTP	725	HTTP/1.1 200 OK (application/dash+xml)
18:36:15.460733000	192.168.1.173	46.31.56.17	HTTP	347	GET /Harmonic/DASH/testmulti/test.mpd HTTP/1.1
18:36:15.462697000	46.31.56.17	192.168.1.173	HTTP	725	HTTP/1.1 200 OK (application/dash+xml)
18:36:15.471428000	192.168.1.173	46.31.56.17	HTTP	347	GET /Harmonic/DASH/testmulti/test.mpd HTTP/1.1
18:36:15.474545000	46.31.56.17	192.168.1.173	HTTP	725	HTTP/1.1 200 OK (application/dash+xml)
18:36:15.482095000	192.168.1.173	46.31.56.17	HTTP	347	GET /Harmonic/DASH/testmulti/test.mpd HTTP/1.1
18:36:15.484036000	46.31.56.17	192.168.1.173	HTTP	725	HTTP/1.1 200 OK (application/dash+xml)
18:36:17.402858000	46.31.56.17	192.168.1.173	HTTP	1238	HTTP/1.1 200 OK (video/mp4)
18:36:17.413333000	192.168.1.173	46.31.56.17	HTTP	374	GET /Harmonic/DASH/testmulti/Stream_0_5400000_Segment-264788.m4v HTTP/1.1
18:36:17.482814000	192.168.1.173	46.31.56.17	HTTP	373	GET /Harmonic/DASH/testmulti/Stream_4_192000_Segment-264788.m4a HTTP/1.1
18:36:17.549300000	46.31.56.17	192.168.1.173	HTTP	1415	HTTP/1.1 200 OK (video/mp4)
18:36:18.198230000	46.31.56.17	192.168.1.173	HTTP	440	HTTP/1.1 200 OK (video/mp4)
18:36:18.207307000	192.168.1.173	46.31.56.17	HTTP	374	GET /Harmonic/DASH/testmulti/Stream_0_5400000_Segment-264789.m4v HTTP/1.1
18:36:18.281107000	192.168.1.173	46.31.56.17	HTTP	373	GET /Harmonic/DASH/testmulti/Stream_4_192000_Segment-264789.m4a HTTP/1.1
18:36:18.305903000	192.168.1.173	46.31.56.17	HTTP	347	GET /Harmonic/DASH/testmulti/test.mpd HTTP/1.1
18:36:18.308592000	46.31.56.17	192.168.1.173	HTTP	725	HTTP/1.1 200 OK (application/dash+xml)
18:36:18.335768000	46.31.56.17	192.168.1.173	HTTP	859	HTTP/1.1 200 OK (video/mp4)
18:36:19.169674000	46.31.56.17	192.168.1.173	HTTP	764	HTTP/1.1 200 OK (video/mp4)
18:36:19.177953000	192.168.1.173	46.31.56.17	HTTP	374	GET /Harmonic/DASH/testmulti/Stream_0_5400000_Segment-264790.m4v HTTP/1.1
18:36:19.228047000	192.168.1.173	46.31.56.17	HTTP	373	GET /Harmonic/DASH/testmulti/Stream_4_192000_Segment-264790.m4a HTTP/1.1
18:36:19.276888000	46.31.56.17	192.168.1.173	HTTP	261	HTTP/1.1 200 OK (video/mp4)
18:36:20.221679000	46.31.56.17	192.168.1.173	HTTP	216	HTTP/1.1 200 OK (video/mp4)
18:36:20.227972000	192.168.1.173	46.31.56.17	HTTP	374	GET /Harmonic/DASH/testmulti/Stream_0_5400000_Segment-264791.m4v HTTP/1.1
18:36:20.273805000	192.168.1.173	46.31.56.17	HTTP	373	GET /Harmonic/DASH/testmulti/Stream_4_192000_Segment-264791.m4a HTTP/1.1
18:36:20.318970000	46.31.56.17	192.168.1.173	HTTP	849	HTTP/1.1 200 OK (video/mp4)

Figura 81. Captura 3 MPEG-DASH con la propiedad de pruebas

Al introducir un corte de 30 segundos, el efecto aumenta de forma notable con unos 10 segundos de corte en la señal. Por último, al aplicar un corte de 35 segundos, desde las 18:42:40 hasta las 18:43:15, el reproductor ya es incapaz de continuar con la reproducción de vídeo teniendo el usuario que reiniciar manualmente la misma. En la captura podemos ver como el player se queda solicitando el siguiente segmento de vídeo pero no se recibe la respuesta, por lo que el reproductor no es capaz de seguir reproduciendo la señal de vídeo. También se puede ver que se piden segmentos sólo de audio pero el reproductor ya no reproduce ni audio ni vídeo.

Time	Source	Destination	Protocol	Length	Info
18:42:34.233986000	192.168.1.173	46.31.56.17	HTTP	347	GET /Harmonic/DASH/testmulti/test.mpd HTTP/1.1
18:42:34.239044000	46.31.56.17	192.168.1.173	HTTP	725	HTTP/1.1 200 OK (application/dash+xml)
18:42:34.400611000	192.168.1.173	46.31.56.17	HTTP	374	GET /Harmonic/DASH/testmulti/Stream_0_5400000_Segment-264868.m4v HTTP/1.1
18:42:34.450707000	192.168.1.173	46.31.56.17	HTTP	373	GET /Harmonic/DASH/testmulti/Stream_4_192000_Segment-264868.m4a HTTP/1.1
18:42:34.491063000	46.31.56.17	192.168.1.173	HTTP	381	HTTP/1.1 200 OK (video/mp4)
18:42:37.708232000	46.31.56.17	192.168.1.173	HTTP	456	HTTP/1.1 200 OK (video/mp4)
18:42:39.220276000	192.168.1.173	46.31.56.17	HTTP	374	GET /Harmonic/DASH/testmulti/Stream_0_5400000_Segment-264869.m4v HTTP/1.1
18:42:39.284672000	192.168.1.173	46.31.56.17	HTTP	373	GET /Harmonic/DASH/testmulti/Stream_4_192000_Segment-264869.m4a HTTP/1.1
18:42:39.331346000	46.31.56.17	192.168.1.173	HTTP	855	HTTP/1.1 200 OK (video/mp4)
18:42:39.346474000	192.168.1.173	46.31.56.17	HTTP	347	GET /Harmonic/DASH/testmulti/test.mpd HTTP/1.1
18:42:39.353944000	46.31.56.17	192.168.1.173	HTTP	725	HTTP/1.1 200 OK (application/dash+xml)
18:43:15.679924000	192.168.1.173	46.31.56.17	HTTP	374	GET /Harmonic/DASH/testmulti/Stream_0_5400000_Segment-264869.m4v HTTP/1.1
18:43:15.925252000	192.168.1.173	46.31.56.17	HTTP	399	GET /Harmonic/DASH/testmulti/Stream_0_5400000_Segment-264869.m4v HTTP/1.1
18:43:15.931789000	192.168.1.173	46.31.56.17	HTTP	399	GET /Harmonic/DASH/testmulti/Stream_0_5400000_Segment-264869.m4v HTTP/1.1
18:43:15.968589000	192.168.1.173	46.31.56.17	HTTP	399	GET /Harmonic/DASH/testmulti/Stream_0_5400000_Segment-264869.m4v HTTP/1.1
18:43:15.975077000	192.168.1.173	46.31.56.17	HTTP	399	GET /Harmonic/DASH/testmulti/Stream_0_5400000_Segment-264869.m4v HTTP/1.1
18:43:15.985602000	192.168.1.173	46.31.56.17	HTTP	373	GET /Harmonic/DASH/testmulti/Stream_4_192000_Segment-264870.m4a HTTP/1.1
18:43:16.058525000	46.31.56.17	192.168.1.173	HTTP	503	HTTP/1.1 200 OK (video/mp4)
18:43:16.067527000	192.168.1.173	46.31.56.17	HTTP	373	GET /Harmonic/DASH/testmulti/Stream_4_192000_Segment-264871.m4a HTTP/1.1
18:43:16.121669000	46.31.56.17	192.168.1.173	HTTP	415	HTTP/1.1 200 OK (video/mp4)
18:43:16.128914000	192.168.1.173	46.31.56.17	HTTP	373	GET /Harmonic/DASH/testmulti/Stream_4_192000_Segment-264872.m4a HTTP/1.1
18:43:16.176885000	46.31.56.17	192.168.1.173	HTTP	349	HTTP/1.1 200 OK (video/mp4)
18:43:16.188202000	192.168.1.173	46.31.56.17	HTTP	373	GET /Harmonic/DASH/testmulti/Stream_4_192000_Segment-264873.m4a HTTP/1.1
18:43:16.234720000	46.31.56.17	192.168.1.173	HTTP	377	HTTP/1.1 200 OK (video/mp4)
18:43:16.241874000	192.168.1.173	46.31.56.17	HTTP	373	GET /Harmonic/DASH/testmulti/Stream_4_192000_Segment-264874.m4a HTTP/1.1
18:43:16.286959000	46.31.56.17	192.168.1.173	HTTP	738	HTTP/1.1 200 OK (video/mp4)
18:43:16.293439000	192.168.1.173	46.31.56.17	HTTP	373	GET /Harmonic/DASH/testmulti/Stream_4_192000_Segment-264875.m4a HTTP/1.1
18:43:16.337449000	46.31.56.17	192.168.1.173	HTTP	489	HTTP/1.1 200 OK (video/mp4)
18:43:16.345698000	192.168.1.173	46.31.56.17	HTTP	373	GET /Harmonic/DASH/testmulti/Stream_4_192000_Segment-264876.m4a HTTP/1.1
18:43:16.392784000	46.31.56.17	192.168.1.173	HTTP	365	HTTP/1.1 200 OK (video/mp4)

Figura 82. Captura 4 MPEG-DASH con la propiedad de pruebas

No obstante, las pruebas realizadas nos muestran que el protocolo es más robusto ante cortes (de un máximo de 25 segundos) que protocolos como HDS o Smooth Streaming.

3.3.4.2. Pruebas MPEG-DASH cambiando la duración del segmento

Como vimos en la parte teórica, MPEG-DASH define el tamaño del segmento como flexible, es decir, no especifica un valor estándar. Por último, y tal y como hemos hecho para el resto de protocolos, vamos a cambiar la duración del segmento y realizaremos una batería de pruebas para ver el comportamiento del protocolo y del player que utilizamos.

Vamos a llevar a cabo las pruebas utilizando el mismo escenario que en el caso anterior con la aplicación Web Browser de la TV conectada de Panasonic.

En primer lugar cambiamos la duración del segmento pasando de los 5 segundos anteriores a una duración de 10 segundos. Para ello accedemos al codificador (equipo Harmonic) y modificamos el tamaño del segmento. Aplicando los cambios podemos acceder al archivo .mpd para ver que se han efectuado correctamente.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <MPD profiles="urn:mpeg:dash:profile:isoff-live:2011" type="dynamic" availabilityStartTime="2014-07-22T15:47:59Z"
    minimumUpdatePeriod="PT10S" minBufferTime="PT1S" timeShiftBufferDepth="PT5M" xmlns="urn:mpeg:DASH:schema:MPD:2011"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="urn:mpeg:DASH:schema:MPD:2011 DASH-MPD.xsd">
  - <Period start="PT0S">
    - <AdaptationSet mimeType="video/mp4" startWithSAP="1" segmentAlignment="true">
        <SegmentTemplate presentationTimeOffset="11" duration="10" startNumber="1" media="$RepresentationID$_Segment-$Number$.m4v"
          initialization="$RepresentationID$_init.m4i" />
        <Representation width="1024" height="576" frameRate="25" codecs="avc1.64001f" scanType="progressive" id="Stream_0_1080000"
          bandwidth="1080000" />
        <Representation width="640" height="360" frameRate="25" codecs="avc1.64001f" scanType="progressive" id="Stream_1_540000"
          bandwidth="540000" />
      </AdaptationSet>
    - <AdaptationSet mimeType="audio/mp4" startWithSAP="1" lang="deu" segmentAlignment="true">
        <SegmentTemplate presentationTimeOffset="11" duration="10" startNumber="1" media="$RepresentationID$_Segment-$Number$.m4a"
          initialization="$RepresentationID$_init.m4i" />
        <Representation audioSamplingRate="48000" codecs="mp4a.40.2" id="Stream_2_192000" bandwidth="192000" />
      </AdaptationSet>
    </Period>
  </MPD>
```

Figura 83. Archivo .mpd MPEG-DASH cambiando la duración del segmento

Cabe destacar que hemos reducido a dos las calidades del stream en formato MPEG-DASH, esto fue necesario para reducir la carga en el AOS debido a que había otras propiedades en servicio. De esta forma nos aseguramos el correcto funcionamiento de esta propiedad.

En primer lugar realizamos una captura del tráfico al iniciar la reproducción en la aplicación Web Browser de la televisión.

Time	Source	Destination	Protocol	Length	Info
17:43:59.366377000	192.168.1.160	46.31.56.17	HTTP	347	GET /Harmonic/DASH/testmulti/test.mpd HTTP/1.1
17:43:59.369172000	46.31.56.17	192.168.1.160	HTTP	436	HTTP/1.1 200 OK (application/dash+xml)
17:43:59.475618000	192.168.1.160	46.31.56.17	HTTP	364	GET /Harmonic/DASH/testmulti/Stream_0_1080000_init.m4i HTTP/1.1
17:43:59.480190000	46.31.56.17	192.168.1.160	HTTP	1144	HTTP/1.1 200 OK (video/mp4)
17:43:59.487313000	192.168.1.160	46.31.56.17	HTTP	372	GET /Harmonic/DASH/testmulti/Stream_0_1080000_Segment-8592.m4v HTTP/1.1
17:43:59.539281000	192.168.1.160	46.31.56.17	HTTP	363	GET /Harmonic/DASH/testmulti/Stream_2_192000_init.m4i HTTP/1.1
17:43:59.547201000	46.31.56.17	192.168.1.160	HTTP	1064	HTTP/1.1 200 OK (video/mp4)
17:43:59.554401000	192.168.1.160	46.31.56.17	HTTP	371	GET /Harmonic/DASH/testmulti/Stream_2_192000_Segment-8592.m4a HTTP/1.1
17:43:59.754964000	46.31.56.17	192.168.1.160	HTTP	1329	HTTP/1.1 200 OK (video/mp4)
17:43:59.808154000	46.31.56.17	192.168.1.160	HTTP	1306	HTTP/1.1 200 OK (video/mp4)
17:43:59.821445000	192.168.1.160	46.31.56.17	HTTP	372	GET /Harmonic/DASH/testmulti/Stream_0_1080000_Segment-8593.m4v HTTP/1.1
17:43:59.882899000	192.168.1.160	46.31.56.17	HTTP	371	GET /Harmonic/DASH/testmulti/Stream_2_192000_Segment-8593.m4a HTTP/1.1
17:44:00.130114000	46.31.56.17	192.168.1.160	HTTP	563	HTTP/1.1 200 OK (video/mp4)
17:44:00.136020000	46.31.56.17	192.168.1.160	HTTP	691	HTTP/1.1 200 OK (video/mp4)
17:44:00.154818000	192.168.1.160	46.31.56.17	HTTP	371	GET /Harmonic/DASH/testmulti/Stream_2_192000_Segment-8594.m4a HTTP/1.1
17:44:00.205442000	192.168.1.160	46.31.56.17	HTTP	372	GET /Harmonic/DASH/testmulti/Stream_0_1080000_Segment-8594.m4v HTTP/1.1
17:44:00.480093000	46.31.56.17	192.168.1.160	HTTP	771	HTTP/1.1 200 OK (video/mp4)
17:44:00.530373000	46.31.56.17	192.168.1.160	HTTP	1189	HTTP/1.1 200 OK (video/mp4)
17:44:08.653319000	192.168.1.160	46.31.56.17	HTTP	347	GET /Harmonic/DASH/testmulti/test.mpd HTTP/1.1
17:44:08.655399000	46.31.56.17	192.168.1.160	HTTP	436	HTTP/1.1 200 OK (application/dash+xml)
17:44:08.696913000	192.168.1.160	46.31.56.17	HTTP	372	GET /Harmonic/DASH/testmulti/Stream_0_1080000_Segment-8595.m4v HTTP/1.1
17:44:08.749793000	192.168.1.160	46.31.56.17	HTTP	371	GET /Harmonic/DASH/testmulti/Stream_2_192000_Segment-8595.m4a HTTP/1.1
17:44:08.884762000	46.31.56.17	192.168.1.160	HTTP	1434	[TCP Out-Of-Order] HTTP/1.1 200 OK (video/mp4)
17:44:08.939803000	46.31.56.17	192.168.1.160	HTTP	128	HTTP/1.1 200 OK (video/mp4)
17:44:18.637300000	192.168.1.160	46.31.56.17	HTTP	371	GET /Harmonic/DASH/testmulti/Stream_2_192000_Segment-8596.m4a HTTP/1.1
17:44:18.697150000	192.168.1.160	46.31.56.17	HTTP	372	GET /Harmonic/DASH/testmulti/Stream_0_1080000_Segment-8596.m4v HTTP/1.1
17:44:18.869166000	192.168.1.160	46.31.56.17	HTTP	347	GET /Harmonic/DASH/testmulti/test.mpd HTTP/1.1
17:44:18.871576000	46.31.56.17	192.168.1.160	HTTP	436	HTTP/1.1 200 OK (application/dash+xml)
17:44:18.896982000	46.31.56.17	192.168.1.160	HTTP	242	HTTP/1.1 200 OK (video/mp4)

Figura 84. Captura 1 MPEG-DASH cambiando la duración del segmento

Podemos ver que al iniciar la reproducción se solicitan 3 segmentos (audio + vídeo) en apenas 1 segundo (alguno menos que en el caso anterior para MPEG-DASH, lo que indica un buffering inicial más lento); entre el tercer y el cuarto segmento hay una duración de casi 9 segundos. A partir de ese momento, se estabiliza la comunicación y se realiza la petición de los segmentos cada 10 segundos (duración del segmento).

Posteriormente, realizamos un corte prolongado con el dispositivo Albedo para analizar el tamaño del buffer del player utilizado. Hay que volver a destacar que se obtiene un

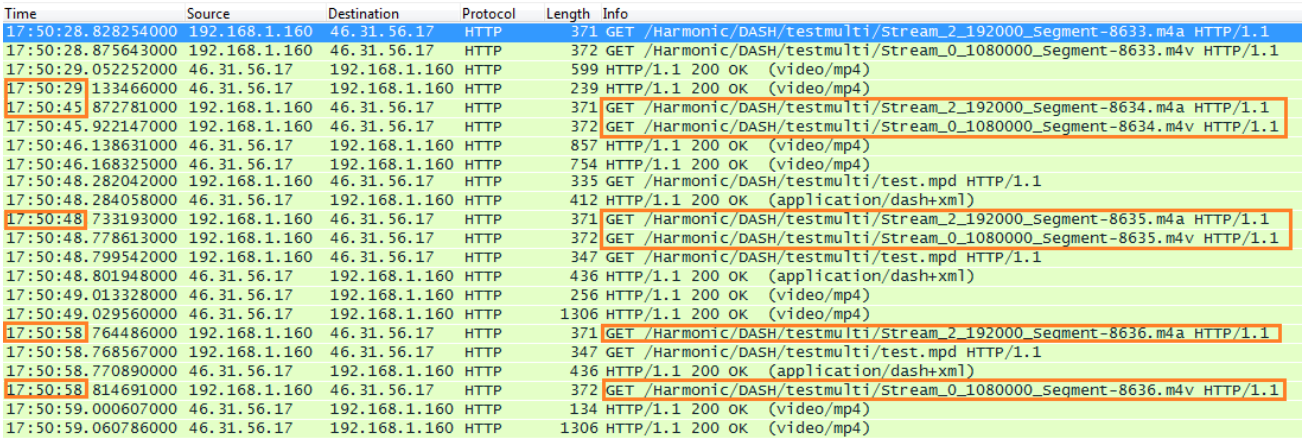
valor aproximado, ya que las medidas pueden depender del momento en el que se efectúa el corte y de factores externos que no controlamos.

A partir del corte, el vídeo continúa durante unos 20 segundos o algo más, lo que equivale a unos 2/3 segmentos.

Cuando la duración del segmento era de 5 segundos (caso anterior) en el buffer se guardaban unos 5 segmentos, por lo que el tiempo de corte ha resultado similar en ambos casos. Varía el número de segmentos debido a la duración de los mismos.

A continuación, realizamos una batería de pruebas introduciendo cortes aumentando de forma progresiva la duración de los mismos.

Al provocar un corte de 10 segundos (tiempo equivalente a la duración de un segmento), desde las 17:50:35 a las 17:50:45, no se aprecia ninguna anomalía en el reproductor de la aplicación que estamos utilizando.



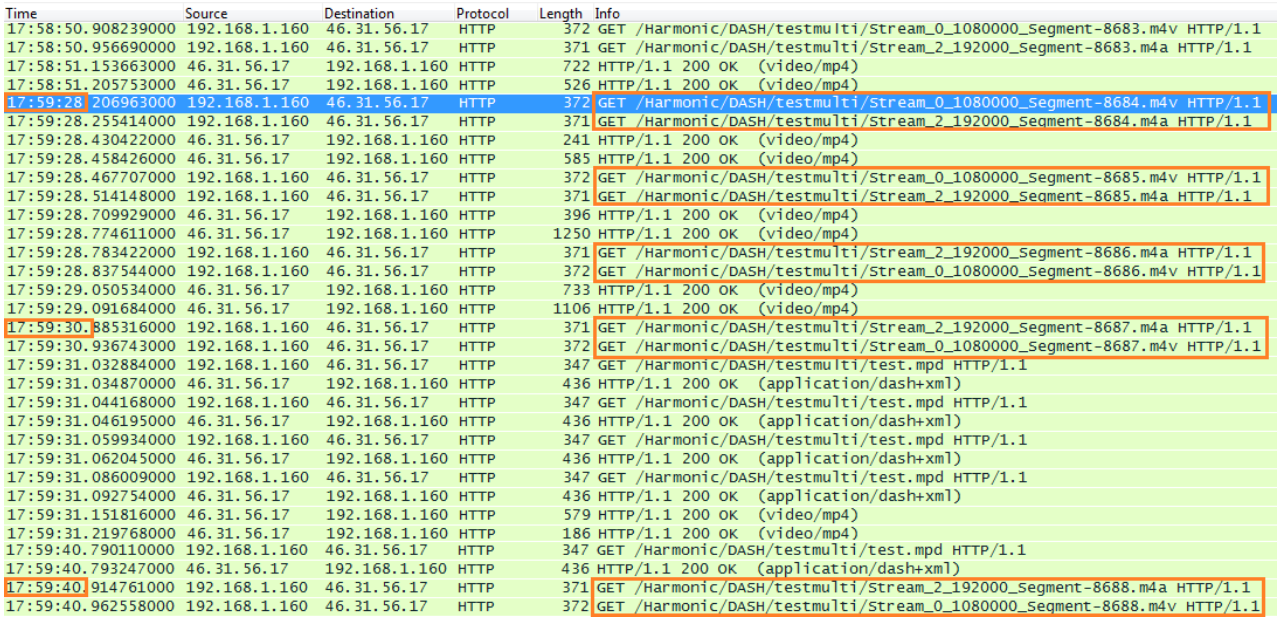
Time	Source	Destination	Protocol	Length	Info
17:50:28.828254000	192.168.1.160	46.31.56.17	HTTP	371	GET /Harmonic/DASH/testmulti/Stream_2_192000_Segment-8633.m4a HTTP/1.1
17:50:28.875643000	192.168.1.160	46.31.56.17	HTTP	372	GET /Harmonic/DASH/testmulti/Stream_0_1080000_Segment-8633.m4v HTTP/1.1
17:50:29.052252000	46.31.56.17	192.168.1.160	HTTP	599	HTTP/1.1 200 OK (video/mp4)
17:50:29.133466000	46.31.56.17	192.168.1.160	HTTP	239	HTTP/1.1 200 OK (video/mp4)
17:50:45.872781000	192.168.1.160	46.31.56.17	HTTP	371	GET /Harmonic/DASH/testmulti/Stream_2_192000_Segment-8634.m4a HTTP/1.1
17:50:45.922147000	192.168.1.160	46.31.56.17	HTTP	372	GET /Harmonic/DASH/testmulti/Stream_0_1080000_Segment-8634.m4v HTTP/1.1
17:50:46.138631000	46.31.56.17	192.168.1.160	HTTP	857	HTTP/1.1 200 OK (video/mp4)
17:50:46.168325000	46.31.56.17	192.168.1.160	HTTP	754	HTTP/1.1 200 OK (video/mp4)
17:50:48.282042000	192.168.1.160	46.31.56.17	HTTP	335	GET /Harmonic/DASH/testmulti/test.mpd HTTP/1.1
17:50:48.284058000	46.31.56.17	192.168.1.160	HTTP	412	HTTP/1.1 200 OK (application/dash+xml)
17:50:48.733193000	192.168.1.160	46.31.56.17	HTTP	371	GET /Harmonic/DASH/testmulti/Stream_2_192000_Segment-8635.m4a HTTP/1.1
17:50:48.778613000	192.168.1.160	46.31.56.17	HTTP	372	GET /Harmonic/DASH/testmulti/Stream_0_1080000_Segment-8635.m4v HTTP/1.1
17:50:48.799542000	192.168.1.160	46.31.56.17	HTTP	347	GET /Harmonic/DASH/testmulti/test.mpd HTTP/1.1
17:50:48.801948000	46.31.56.17	192.168.1.160	HTTP	436	HTTP/1.1 200 OK (application/dash+xml)
17:50:49.013328000	46.31.56.17	192.168.1.160	HTTP	256	HTTP/1.1 200 OK (video/mp4)
17:50:49.029560000	46.31.56.17	192.168.1.160	HTTP	1306	HTTP/1.1 200 OK (video/mp4)
17:50:58.764486000	192.168.1.160	46.31.56.17	HTTP	371	GET /Harmonic/DASH/testmulti/Stream_2_192000_Segment-8636.m4a HTTP/1.1
17:50:58.768567000	192.168.1.160	46.31.56.17	HTTP	347	GET /Harmonic/DASH/testmulti/test.mpd HTTP/1.1
17:50:58.770890000	46.31.56.17	192.168.1.160	HTTP	436	HTTP/1.1 200 OK (application/dash+xml)
17:50:58.814691000	192.168.1.160	46.31.56.17	HTTP	372	GET /Harmonic/DASH/testmulti/Stream_0_1080000_Segment-8636.m4v HTTP/1.1
17:50:59.000607000	46.31.56.17	192.168.1.160	HTTP	134	HTTP/1.1 200 OK (video/mp4)
17:50:59.060786000	46.31.56.17	192.168.1.160	HTTP	1306	HTTP/1.1 200 OK (video/mp4)

Figura 85. Captura 2 MPEG-DASH cambiando la duración del segmento

Podemos ver que nada más recuperar la conexión se solicitan los segmentos de vídeo y audio, y a los 3 segundos se vuelve a solicitar un nuevo segmento. El siguiente segmento ya se solicita pasados 10 segundos (volviendo a la normalidad).

Si realizamos un corte de 20 segundos, la reproducción tampoco se para. El comportamiento es similar al anterior, al recuperar la conexión se solicitan 3 segmentos en 6 segundos, volviendo posteriormente a los cauces habituales.

Al realizar un corte de 25 segundos, desde las 17:59:00 a las 17:59:25, la reproducción se ve interrumpida, es decir, ya hay afectación del servicio cara al usuario. El corte en la imagen tiene una duración de unos 8/9 segundos aproximadamente (aunque puede variar en función del instante inicial del corte que causemos).



Time	Source	Destination	Protocol	Length	Info
17:58:50.908239000	192.168.1.160	46.31.56.17	HTTP	372	GET /Harmonic/DASH/testmulti/Stream_0_1080000_Segment-8683.m4v HTTP/1.1
17:58:50.956690000	192.168.1.160	46.31.56.17	HTTP	371	GET /Harmonic/DASH/testmulti/Stream_2_192000_Segment-8683.m4a HTTP/1.1
17:58:51.153663000	46.31.56.17	192.168.1.160	HTTP	722	HTTP/1.1 200 OK (video/mp4)
17:58:51.205753000	46.31.56.17	192.168.1.160	HTTP	526	HTTP/1.1 200 OK (video/mp4)
17:59:28.206963000	192.168.1.160	46.31.56.17	HTTP	372	GET /Harmonic/DASH/testmulti/Stream_0_1080000_Segment-8684.m4v HTTP/1.1
17:59:28.255414000	192.168.1.160	46.31.56.17	HTTP	371	GET /Harmonic/DASH/testmulti/Stream_2_192000_Segment-8684.m4a HTTP/1.1
17:59:28.430422000	46.31.56.17	192.168.1.160	HTTP	241	HTTP/1.1 200 OK (video/mp4)
17:59:28.458426000	46.31.56.17	192.168.1.160	HTTP	585	HTTP/1.1 200 OK (video/mp4)
17:59:28.467707000	192.168.1.160	46.31.56.17	HTTP	372	GET /Harmonic/DASH/testmulti/Stream_0_1080000_Segment-8685.m4v HTTP/1.1
17:59:28.514148000	192.168.1.160	46.31.56.17	HTTP	371	GET /Harmonic/DASH/testmulti/Stream_2_192000_Segment-8685.m4a HTTP/1.1
17:59:28.709929000	46.31.56.17	192.168.1.160	HTTP	396	HTTP/1.1 200 OK (video/mp4)
17:59:28.774611000	46.31.56.17	192.168.1.160	HTTP	1250	HTTP/1.1 200 OK (video/mp4)
17:59:28.783422000	192.168.1.160	46.31.56.17	HTTP	371	GET /Harmonic/DASH/testmulti/Stream_2_192000_Segment-8686.m4a HTTP/1.1
17:59:28.837544000	192.168.1.160	46.31.56.17	HTTP	372	GET /Harmonic/DASH/testmulti/Stream_0_1080000_Segment-8686.m4v HTTP/1.1
17:59:29.050534000	46.31.56.17	192.168.1.160	HTTP	733	HTTP/1.1 200 OK (video/mp4)
17:59:29.091684000	46.31.56.17	192.168.1.160	HTTP	1106	HTTP/1.1 200 OK (video/mp4)
17:59:30.885316000	192.168.1.160	46.31.56.17	HTTP	371	GET /Harmonic/DASH/testmulti/Stream_2_192000_Segment-8687.m4a HTTP/1.1
17:59:30.936743000	192.168.1.160	46.31.56.17	HTTP	372	GET /Harmonic/DASH/testmulti/Stream_0_1080000_Segment-8687.m4v HTTP/1.1
17:59:31.032884000	192.168.1.160	46.31.56.17	HTTP	347	GET /Harmonic/DASH/testmulti/test.mpd HTTP/1.1
17:59:31.034870000	46.31.56.17	192.168.1.160	HTTP	436	HTTP/1.1 200 OK (application/dash+xml)
17:59:31.044168000	192.168.1.160	46.31.56.17	HTTP	347	GET /Harmonic/DASH/testmulti/test.mpd HTTP/1.1
17:59:31.046195000	46.31.56.17	192.168.1.160	HTTP	436	HTTP/1.1 200 OK (application/dash+xml)
17:59:31.059934000	192.168.1.160	46.31.56.17	HTTP	347	GET /Harmonic/DASH/testmulti/test.mpd HTTP/1.1
17:59:31.062045000	46.31.56.17	192.168.1.160	HTTP	436	HTTP/1.1 200 OK (application/dash+xml)
17:59:31.086009000	192.168.1.160	46.31.56.17	HTTP	347	GET /Harmonic/DASH/testmulti/test.mpd HTTP/1.1
17:59:31.092754000	46.31.56.17	192.168.1.160	HTTP	436	HTTP/1.1 200 OK (application/dash+xml)
17:59:31.151816000	46.31.56.17	192.168.1.160	HTTP	579	HTTP/1.1 200 OK (video/mp4)
17:59:31.219768000	46.31.56.17	192.168.1.160	HTTP	186	HTTP/1.1 200 OK (video/mp4)
17:59:40.790110000	192.168.1.160	46.31.56.17	HTTP	347	GET /Harmonic/DASH/testmulti/test.mpd HTTP/1.1
17:59:40.793247000	46.31.56.17	192.168.1.160	HTTP	436	HTTP/1.1 200 OK (application/dash+xml)
17:59:40.914761000	192.168.1.160	46.31.56.17	HTTP	371	GET /Harmonic/DASH/testmulti/Stream_2_192000_Segment-8688.m4a HTTP/1.1
17:59:40.962558000	192.168.1.160	46.31.56.17	HTTP	372	GET /Harmonic/DASH/testmulti/Stream_0_1080000_Segment-8688.m4v HTTP/1.1

Figura 86. Captura 3 MPEG-DASH cambiando la duración del segmento

Vemos que al recuperar la conexión se envían 4 segmentos (audio + vídeo) en un intervalo de 3 segundos. En la captura también se pueden ver varias peticiones del archivo .mpd.

Al realizar un corte de 30 segundos, obtuvimos resultados semejantes al caso anterior, con cortes en torno a los 8/9 segundos en la reproducción. Hay que destacar que la diferencia de tiempo entre un corte y otro (corte de 25 segundos) era la mitad de la duración de un segmento, lo que justifica que no haya grandes diferencias.

Al provocar un corte de 40 segundos (incluso 35 segundos), el vídeo se cortaba y la reproducción no se recuperaba. La única forma de reiniciarla era manualmente.

Hay que destacar que realizando las pruebas nos hemos encontrado veces en las que al provocar un corte de la señal, el reproductor no ha conseguido recuperar la reproducción pero esto es debido a factores que no podemos controlar, por lo que las medidas/pruebas son variables (hemos intentado reflejar los resultados de la forma más veraz posible).

MPEG-DASH			
PLATAFORMA/PLAYER	**TAMAÑO DEL BUFFER**	**DURACIÓN CHUNKS**	**MAXIMO CORTE SOPORTADO**
Aplicación TV Conectada	25 segundos	5 segundos	35 segundos
Aplicación TV Conectada	20 segundos	10 segundos	35 segundos

Tabla 8. Resumen pruebas MPEG-DASH

En este caso el cambiar la duración del segmento no ha provocado efectos considerables. Como ya vimos en la parte teórica, el tener segmentos más largos o más

cortos tiene sus ventajas dependiendo del uso que se quiera dar, pero en este caso el codificador empleado nos restringe a utilizar una duración del segmento entre 2 y 10 segundos.

Al aumentar la duración del segmento, el tamaño del buffer ha disminuido ligeramente. Por otra parte, en ambos casos coincide en que el máximo corte soportado, que no obliga a reiniciar manualmente la aplicación, es el mismo. Ante cortes superiores a 35 segundos hay que reiniciar la aplicación de forma manual.

4. Conclusiones y líneas futuras

4.1. Conclusiones

Las pruebas realizadas nos han permitido ver cómo se comporta cada protocolo en función de los players utilizados. Hemos podido analizar la robustez de los distintos protocolos ante cortes en la señal de vídeo recibida y cómo influye utilizar distintas duraciones de los segmentos o fragmentos en cada uno de los protocolos.

HLS			
PLATAFORMA/PLAYER	**TAMAÑO DEL BUFFER**	**DURACIÓN CHUNKS**	**MAXIMO CORTE SOPORTADO**
Nubeox	15/18 segundos	8 segundos	20/24 segundos
Ono	20/30 segundos	10 segundos	20/24 segundos
Safari	20 segundos	10 segundos	< 30 segundos
Safari	12 segundos	4 segundos	16 segundos
SMOOTH STREAMING			
PLATAFORMA/PLAYER	**TAMAÑO DEL BUFFER**	**DURACIÓN CHUNKS**	**MAXIMO CORTE SOPORTADO**
Ono	6 segundos	2 segundos	38 segundos
Silverlight	8 segundos	2 segundos	22 segundos
Silverlight	---	8 segundos	---
HDS			
PLATAFORMA/PLAYER	**TAMAÑO DEL BUFFER**	**DURACIÓN CHUNKS**	**MAXIMO CORTE SOPORTADO**
Nubeox	6 segundos	4 segundos	+ 50 segundos
Reproductor CDN	7.7-12 segundos	4 segundos	+28 segundos
Reproductor CDN	7.7-14 segundos	8 segundos	+ 40 segundos
MPEG-DASH			
PLATAFORMA/PLAYER	**TAMAÑO DEL BUFFER**	**DURACIÓN CHUNKS**	**MAXIMO CORTE SOPORTADO**
Aplicación TV Conectada	25 segundos	5 segundos	35 segundos
Aplicación TV Conectada	20 segundos	10 segundos	35 segundos

Tabla 9. Resumen comparativa de protocolos

En la tabla anterior recogemos los resultados de las pruebas realizadas.

Por un lado tenemos el tamaño del buffer que nos da una idea de la robustez del protocolo ante cortes en la señal de vídeo. Además, se indica la duración de los chunks empleada en las distintas pruebas. En la última columna indicamos el tiempo del corte introducido que a pesar de provocar cortes en la señal de vídeo reproducida en el player, hace que la reproducción se recupere de manera automática (sin tener que reiniciarla manualmente). Este tiempo no representa que el protocolo sea más robusto ante imperfecciones/cortes en la señal, sino que indica una característica del player menos importante que el tamaño del buffer; ya que en condiciones óptimas el objetivo es que no se produzcan grandes cortes en la señal de vídeo.

Analizando los resultados, podemos concluir que el protocolo HLS tiene un tamaño de buffer mayor que HDS y Smooth Streaming, lo que le hace más resistente a cortes (no sufrir ninguna alteración en la reproducción cara al usuario). Por otro lado, al ser los segmentos de mayor duración en HLS (normalmente 10 segundos) el buffering inicial es más lento que el de HDS y Smooth Streaming. También es el protocolo en el que se realizan más descargas del sub-manifest.

MPEG-DASH es un protocolo en evolución y en las pruebas realizadas hemos visto que es el protocolo más resistente a los cortes, llegando a no tener ningún tipo de interrupción en el vídeo ante cortes de 25 segundos.

En Smooth Streaming, al ser la duración de los fragmentos de 2 segundos, hay un buffering inicial bastante rápido; por otro lado es el protocolo que presenta un buffer más pequeño. Al probar y aumentar la duración del segmento, vimos como el reproductor estándar falla y se para automáticamente.

En HDS, el tamaño del buffer es superior al de Smooth Streaming. En los reproductores utilizados con este protocolo, a pesar de introducir cortes bastante largos, se ha recuperado la reproducción de forma automática, ofreciendo en este caso mejores prestaciones que en los protocolos HLS o Smooth Streaming.

En MPEG-DASH no hemos notado grandes diferencias al cambiar el tamaño del segmento. Con una duración de 10 segundos del tamaño del chunk, el buffer disminuía ligeramente. También es cierto que el codificador sólo nos permitía fijar duraciones para los segmentos entre 2 segundos y 10 segundos, por lo que no pudimos probar con una duración mayor (que a priori nos permite este protocolo).

Otra conclusión que obtenemos es que cuando se comienza una reproducción siempre se suele comenzar por la calidad más baja, digamos que siempre hay que suponer que el usuario puede encontrarse en el peor caso (peores condiciones en cuanto a ancho de banda disponible, carga cpu,…). Cuando se comprueba que se puede realizar un cambio a una calidad superior, inmediatamente se lleva a cabo de forma adaptativa, sin provocar ningún tipo de corte. También es cierto que puede haber ciertos reproductores

que estén configurados de forma que comiencen por una determinada calidad que no tiene porque ser la más baja, pero no suele ser el caso habitual.

Por lo comentado anteriormente, es recomendable no incluir perfiles de sólo audio en señales de vídeo, ya que al iniciar la reproducción el player comenzará a reproducir el audio únicamente y tardará más tiempo en presentar el vídeo (algo que no nos interesa).

Además, también es recomendable usar perfiles de bit-rate bajo teniendo en cuenta el estado de las líneas de Internet hoy en día (sin ancho de banda garantizado).

Hay que tener en cuenta la variabilidad de los resultados obtenidos en la realización de las pruebas ya que nunca se obtienen los mismos resultados entre una prueba y otra aunque sean de idénticas características. Factores como la calidad de la línea, el punto de trabajo del algoritmo del player utilizado o variaciones en la entrega del fragmento/segmento derivadas de Internet son causas que afectan a esas variaciones. En consecuencia, el análisis de los resultados siempre tiene que ser bastante cauteloso, ya que esta variabilidad hace que lo que funciona en un momento pueda fallar en otro.

En general, no podemos establecer que haya un protocolo mejor que otro, ya que cada uno está orientado a una tecnología, Smooth Streaming para Microsoft, HLS para dispositivos Apple (iPad, iPhones) y HDS para dispositivos compatibles con Adobe Flash. La que trata realmente de aunar todas las tecnologías es MPEG-DASH y es la que a priori se puede presentar como la solución del futuro, aunque de momento parece que estos protocolos coexistirán durante unos años.

4.2. Líneas futuras

En este proyecto se ha dado un primer paso de acercamiento a los protocolos de transmisión de vídeo adaptativos. Acercándonos por un lado al proveedor de servicios de CDN y por otro al operador de comunicaciones o empresa que ofrece canales de TV, nosotros hemos ejercido como el punto intermedio entre ambos. Al primero es al que solicitamos las propiedades para la transmisión de vídeo on-line y éste nos ofrece sus servidores para la distribución del contenido. A los segundos les podríamos dar soporte para poder ofrecer sus servicios de TV on-line (de hecho es lo que se está comenzando a hacer).

Una línea futura a seguir para este proyecto podría ser centrarnos en uno de los protocolos estudiados, por ejemplo MPEG-DASH que es el más novedoso y tratar de explotar todas sus características y ver la posibilidad de poderlo utilizar en múltiples dispositivos. Para ello hay foros de MPEG-DASH dónde los ingenieros y analistas intentan dar continuidad a este proyecto [20].

Por otro lado, se puede dar continuidad a este proyecto por la vía del mantenimiento o monitorizado de un servicio a una de esas empresas que ofrecen canales de TV online o en nuestro caso con las propiedades que tenemos a nuestra disposición.

Hoy en día, el ofrecer al usuario un buen servicio es muy importante por lo que hay que tratar de minimizar los posibles fallos que pueda haber a la hora de dar una distribución de servicios, en este caso, de vídeo por Internet. Igual que existen centros de control que se encargan de monitorizar los diferentes canales de TV terrestre, tenemos que tratar de tener un sistema de supervisión del contenido que se distribuye a través de una CDN.

Para ello existe un potente sistema de monitorizado denominado Nagios [35] con el que se pueden crear diferentes "checks" a través de scripts programados (en BASH [36]) para el monitorizado del servicio que se esté ofreciendo. Además este programa nos permite crear mapas para visualizar de forma gráfica el esquema de nuestra red.

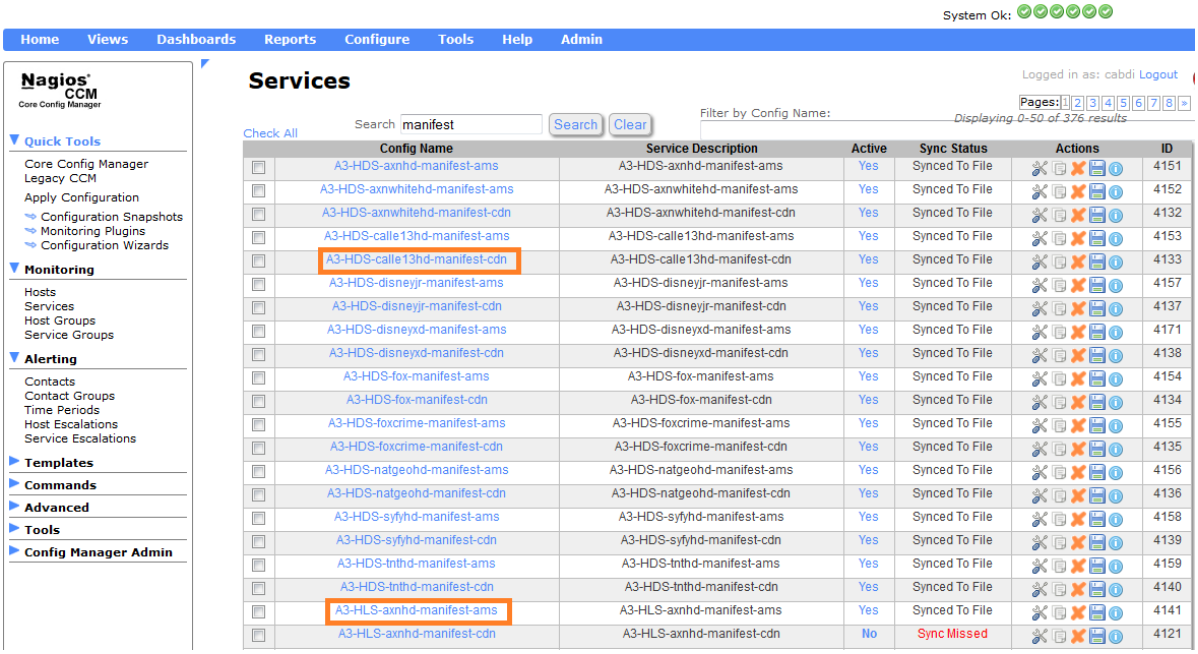


Figura 87. Sistema de monitorado Nagios

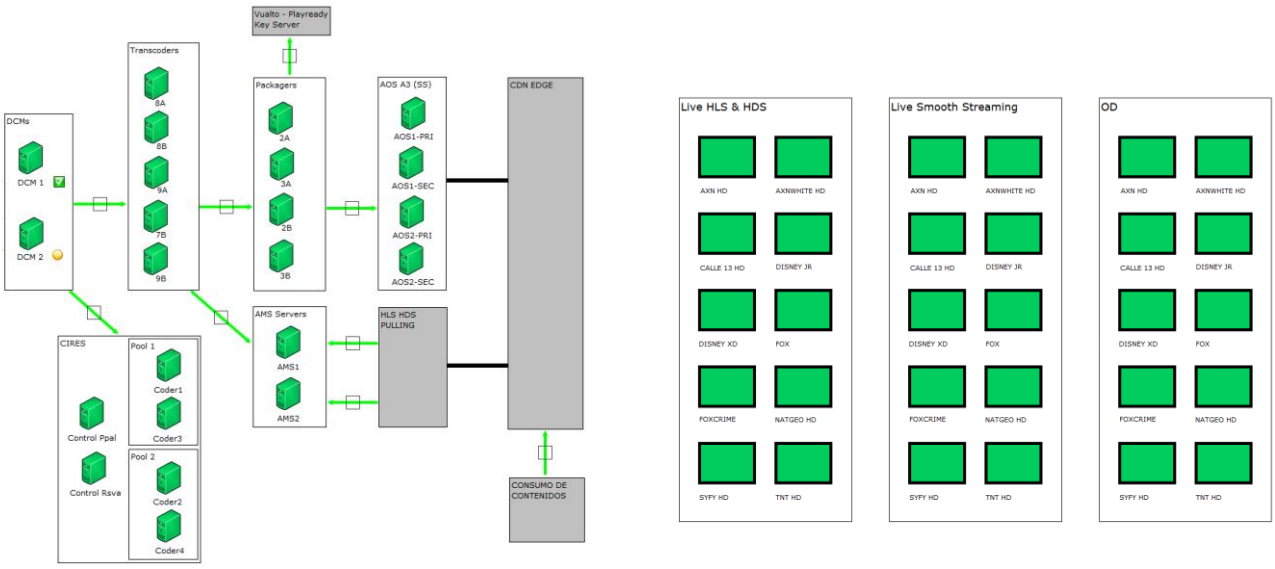


Figura 88. Mapas creados en Nagios

Dentro de un mapa podemos ordenar los canales por protocolos, en el ejemplo anterior los tenemos ordenados por canales Live tanto HLS como HDS, canales Live de Smooth Streaming o canales On Demand. Algunos de los chequeos que se pueden configurar son los de acceso al Manifest de una propiedad o los de acceso a cada uno de los sub-manifest de las diferentes calidades de un contenido de vídeo.

Host	Service	Status	Duration	Attempt	Last Check	Status Information
A3-HLS-LIVE-CDN	A3-HLS-foxcrime-seq_chunk_audio-cdn	Ok	17d 15h 25m 16s	1/1	04/08/2014 08:47:20	OK: TS 10 Seg analysis. 1r Chunk: 974094 - 2o Chunk: 974095, diferencia: 1
	A3-HLS-calle13hd-seq_chunk_MQ-cdn	Ok	9d 8h 55m 41s	1/1	04/08/2014 08:47:17	OK: TS 10 Seg analysis. 1r Chunk: 32974093 - 2o Chunk: 32974094, diferencia: 1
A3-HSS1-LIVE-CDN	A3-Smooth-fox-manifest-cdn	Ok	6d 5h 29m 39s	1/5	04/08/2014 08:47:54	OTT-MON HTTP OK 200 - ok: 0.072 seconds response time
A3-HLS-LIVE-CDN	A3-HLS-disneyxd-seq_chunk_audio-cdn	Ok	6d 0h 12m 3s	1/1	04/08/2014 08:47:26	OK: TS 10 Seg analysis. 1r Chunk: 437223 - 2o Chunk: 437225, diferencia: 2
	A3-HLS-axnwhitehd-seq_chunk_MQ-cdn	Ok	5d 23h 7m 3s	1/1	04/08/2014 08:48:24	OK: TS 10 Seg analysis. 1r Chunk: 437230 - 2o Chunk: 437232, diferencia: 2
	A3-HLS-disneyjr-seq_chunk_LQ-cdn	Ok	5d 23h 6m 59s	1/1	04/08/2014 08:48:01	OK: TS 10 Seg analysis. 1r Chunk: 437228 - 2o Chunk: 437229, diferencia: 1
A3-HDS-LIVE-CDN	A3-HDS-foxcrime-manifest-cdn	Ok	5d 18h 39m 11s	1/5	04/08/2014 08:47:54	OTT-MON HTTP OK 200 - ok: 0.057 seconds response time
A3-HLS-LIVE-CDN	A3-HLS-syfyhd-seq_chunk_HQ-cdn	Ok	5d 16h 32m 43s	1/1	04/08/2014 08:48:01	OK: TS 10 Seg analysis. 1r Chunk: 974099 - 2o Chunk: 974100, diferencia: 1
	A3-HLS-axnhd-seq_chunk_HQ-cdn	Ok	5d 15h 51m 6s	1/1	04/08/2014 08:47:16	OK: TS 10 Seg analysis. 1r Chunk: 437222 - 2o Chunk: 437223, diferencia: 1
A3-HDS-LIVE-CDN	A3-HDS-axnwhitehd-manifest-cdn	Ok	5d 13h 1m 15s	1/5	04/08/2014 08:48:01	OTT-MON HTTP OK 200 - ok: 0.049 seconds response time
A3-HLS-LIVE-CDN	A3-HLS-disneyxd-manifest-cdn	Ok	5d 11h 34m 18s	1/5	04/08/2014 08:47:23	OTT-MON Encriptacion OK - OTT-MON HTTP OK 200 - ok: 0.050 seconds response time
	A3-HLS-foxcrime-seq_chunk_LQ-cdn	Ok	5d 11h 33m 29s	1/1	04/08/2014 08:47:16	OK: TS 10 Seg analysis. 1r Chunk: 974093 - 2o Chunk: 974094, diferencia: 1
	A3-HLS-axnwhitehd-seq_chunk_HQ-cdn	Ok	5d 11h 33m 27s	1/1	04/08/2014 08:48:44	OK: TS 10 Seg analysis. 1r Chunk: 437233 - 2o Chunk: 437234, diferencia: 1

Figura 89. Monitorizado de Nagios

Ésta sería una línea futura para dar continuidad al proyecto y que podría servir para dar un servicio real (como realmente ya se está comenzando a dar).

5. Presupuesto

Para elaborar el presupuesto del proyecto hay que tener en cuenta los distintos equipos y dispositivos utilizados, así como el coste del personal, es decir, las horas de trabajo del ingeniero que se encarga de realizar las pruebas. También habrá que contabilizar lo que cuesta el uso de las propiedades para pruebas solicitadas al proveedor de la CDN.

La duración del proyecto ha sido de ocho meses, dónde se incluye el estudio previo y las horas dedicadas a las pruebas en el laboratorio.

Por tanto, para establecer los costes del personal del proyecto tendremos en cuenta la figura de un ingeniero que ha trabajado durante 8 meses una media de 3 horas diarias, y la figura del director del proyecto. Para el coste por hora del trabajo de ingeniero cogeremos un valor habitual a día de hoy establecido para los proyectos de telecomunicaciones.

Concepto	Coste/hora	Coste total (sin IVA)	Coste total (con 21% IVA)
Ingeniero	39.89 €/hora	19147.2 €	23168.112 €
Director del Proyecto		1531.77 €	1853.44 €
		Total	**25021.552 €**

Tabla 10. Costes del personal

Los equipos que se han utilizado y su coste son los siguientes:

Material	Unidades	Precio (incluye IVA)
ProMedia 1200 ATO HARMONIC	1	24734 €
ALBEDO Net.Shark	1	6625 €
ALBEDO Net.Storm	1	8900 €
Portátil Toshiba	1	565 €
iPad	1	399 €
TV Panasonic	1	385 €
	Total	**41608 €**

Tabla 11. Costes de los equipos

Además hay un coste asociado a las propiedades que hemos utilizado, en particular al uso de la CDN. Han sido 4 propiedades de pruebas (Smooth Streaming, HLS, HDS y

MPEG-DASH), más el acceso como usuarios a las plataformas de Nubeox y Ono durante 4 meses.

Usos	Meses	Precio
CDN	4	605 € (incluido IVA)
Nubeox	4	10 €/mes * 4 meses
Ono	4	31.34 €/mes * 4 meses
	Total	**770.36 €**

Tabla 12. Costes por usos de aplicaciones

El presupuesto total estaría dado por la suma de los tres costes mostrados anteriormente:

Presupuesto Costes Totales	Presupuestos Costes Totales
Personal	25021.552 €
Material	41608 €
Uso CDN/Aplicaciones	770.36 €
Total	**67399.912 €**

Tabla 13. Presupuesto total

Por tanto, el presupuesto total del proyecto asciende a 67399.912 €.

GLOSARIO

AOM Adaptive Origin Manager

AOS Adaptive Origin Server

BASH Bourne Again SHell

CDN Content Delivery Network

DASH Dynamic Adaptive Streaming over HTTP

DRM Digital Rights Management

DVR Digital Video Recorder

FTP File Transfer Protocol

GIF Graphics Interchange Format

HTTP HyperText Transfer Protocol

HE-AAC High-Efficiency Advanced Audio Coding

HE/SBR High Efficiency / Scale Band Replication

HLS HTTP Live Streaming

HDS HTTP Dynamic Streaming

HSS HTTP Smooth Streaming

iOS iPhone Operating System

JPEG Joint Photographics Experts Group

MPD Media Presentation Description

MPEG Moving Picture Experts Group

MP3 Moving Picture Experts Group Layer-3

PNG Portable Network Graphics

RTMP Real Time Messaging Protocol

RTP Real Time Protocol

RTSP Real Time Streaming Protocol

3G	Third Generation
TCP	Transmission Control Protocol
TS	Transport Stream
UDP	User Datagram Protocol
VOD	Video On Demand
WMV	Windows Media Video
XML	eXtensible Markup Language

BIBLIOGRAFÍA

[1] White Paper, "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2013–2018", Cisco, February 2014.

[2] "Reproductor de Silverlight," http://playready.directtaps.net/pr/doc/slee/, Jun 2014.

[3] "Information Technology-Coding of Audio-Visual Objects- Part 12: ISO Base Media File Format", ISO/IEC 14496-12, 2008.

[4] ITU-T Rec. H.222.0|ISO/IEC 13818-1, "Information Technology—Generic Coding of Moving Pictures and Associated Audio Information: Systems", ITU-T/ ISO/IEC, 2007; http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=44169.

[5] ITU-T Rec.H.262 | ISO/IEC 13818-2, "Information technology – Generic coding of moving pictures and associated audio information: Video".

[6] T. Kristensen, P. Luthi: "RTP Payload Format for H.264 Reduced-Complexity Decoding Operation (RCDO) Video", RFC-6185, IETF, May 2011.

[7] Y.-K. Wang, R. Even, Huawei Technologies, T. Kristensen, Tandberg, R. Jesup, WorldGate Communications, "RTP Payload Format for H.264 Video", RFC-6184, IETF, May 2011.

[8] [MS-SSTR] — v20140124 Smooth Streaming Protocol, Thursday, February 2014.

[9] "Delivery Live and On-Demand Streaming Media", Microsoft Silverlight, iStreamPlanet.

[10] "Página oficial de Microsoft, Smooth Streaming," http://www.iis.net/downloads/microsoft/smooth-streaming/, Jul 2014.

[11] "Live and On-Demand Video with Silverlight and IIS Smooth Streaming", Microsoft Corporation, February 2010.

[12] Alex Zambelli, Media Technology Evangelist, "IIS Smooth Streaming Technical Overview", Microsoft Corporation, March 2009.

[13] David M. Nelson, "Smooth Streaming Deployment Guide", http://www.iis.net/learn/media/smooth-streaming/smooth-streaming-deployment-guide/, Microsoft Corporation, August 2010.

[14] R. Pantos, Ed. W. May, Informational Internet Draft "HTTP Live Streaming

draft-pantos-http-live-streaming-12", Apple Inc, October 14, 2013.

[15] "Página oficial Apple HLS," https://developer.apple.com/streaming/, Jul 2014.

[16] “HTTP Dynamic Streaming Specification Version 3.0 FINAL”, Adobe, August 2013.

[17] “Flash ® Media Manifest (F4M) Format Specification Version 3.0 FINAL”, Adobe, August 2013.

[18] “Video File Format Specification Version 10.1,” http://download.macromedia.com/f4v/video_file_format_spec_v10_1.pdf

[19] “The Definitive Guide to HDS,” http://www.encoding.com/http-dynamic-streaming-hds/, Version 1.0, Released: August 9, 2013.

[20] “Página oficial de MPEG-DASH,” http://dashif.org/, Jul 2014.

[21] I. Sodagar, “The MPEG-DASH Standard for Multimedia Streaming Over the Internet”, White Paper, IEEE Multimedia, Oct-Nov 2011.

[22] T. Stockhammer, I. Sodagar, “MPEG DASH: The Enabler Standard for Video Deliver Over The Open Internet,” IBC Conference 2011, Sept 2011.

[23] I. Sodagar and H. Pyle, “Reinventing multimedia delivery with MPEG-DASH”, SPIE Applications of Digital Image Processing XXXIV, Sept 2011.

[24] T. Stockhammer: “Dynamic Adaptive Streaming over HTTP-Design Principles and Standards” In: MMSys ’11: Proceedings of the second annual ACM conference on Multimedia systems New York, NY, USA: ACM Press, Feb 2011, S. 133-14.

[25] ETSI TS 126 247, “Universal Mobile Telecommunications System (UMTS); LTE; Transparent end-to-end Packet-switched Streaming Service (PSS); Progressive Download and Dynamic Adaptive Streaming over HTTP (3GP-DASH) (3GPP TS 26.247 version 10.0.0 Release 10)”.

[26] T. Stockhammer, “MPEG's Dynamic Adaptive Streaming over HTTP (DASH) – Enabling Formats for Video Streaming over the Open Internet”, Qualcomm Incorporated, Webinar at EBU- Nov 22, 2011.

[27] Joel Unickow, “The DASH Handbook”, June/July 2013.

[28] Iraj Sodagar, “Overview of MPEG-DASH ISO/IEC 23009-1”, November 2011.

[29] Ulrich Grönewald, “MPEG-DASH, Dynamic Adaptive Streaming over HTTP. What, When and How?”, Cisco Knowledge Network, January 2013.

[30] White Paper, “Adaptive Bitrate Streaming”, Tektronix, December 2013.

[31] “Página oficial de Whireshark,” http://www.wireshark.org/, Jul 2014.

[32] “Página oficial de Fiddler,” http://www.telerik.com/fiddler/, Jul 2014.

[33] Manual de usuario. “Net.Storm. Network Impairment Generator”. Albedo Telecom.

[34] “Página oficial de Albedo,” http://www.albedotelecom.com/, Jul 2014.

[35] “Página oficial de Nagios,” http://www.nagios.org/, August 2014.

[36] “Tutorial de BASH”, http://linuxconfig.org/bash-scripting-tutorial, August 2014.

[37] Jan Ozer, “Video Compression for Flash, Apple Devices and Html5”, Doceo Publishing, http://www.onlinevideo.net/2011/05/streaming-vs-progressive-download-vs-adaptive-streaming/, August 2014.

[38] “Blog de Verizon Digital Media Services”, http://blog.edgecast.com/post/55198896476/hds-hls-hss-adaptive-http-streaming-demystified/, “ Digital Media and Web Performance”, Jul 11th, 2013.

[39] P. Deutsch, “GZIP file format specification version 4”, RFC-1952, May 1996.